VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ANALÝZA DIAGRAMŮ BYZNYS PROCESŮ

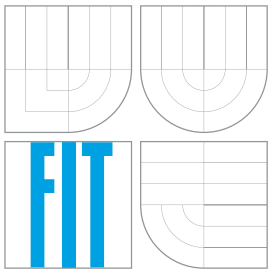
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MARTIN LUDVÍK

BRNO 2009

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ANALÝZA DIAGRAMŮ BYZNYS PROCESŮ

ANALYSIS OF BUSINESS PROCESS DIAGRAMS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MARTIN LUDVÍK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PETR WEISS

BRNO 2009

Abstrakt

Cílem projektu *Analýza diagramů byznys procesů* je vytvořit postup, kterým je možné najít typické vzory v rámci diagramů byznys procesů. Posléze navrhnout a implementovat aplikaci, která bude schopna tyto vzory vyhledat a označit. To vše na základě diagramu uloženého ve formě XML dokumentu. Důležitým prvkem je zachování možnosti rozšíření počtu vyhledávaných vzorů a také snaha o řešení různých nestandardních situací jako například překryv vzorů.

Abstract

The aim of *Analysis of Business Process Diagrams* is to create procedure that is able to find typical patterns in business process diagrams. Besides, it is necessary to design and implement application, which will be able to find and mark selected workflow patterns. This analysis is based on XML document, in which a business process diagram is stored. Important thing is to keep the ability to extend the number of patterns, which can be find out in a diagram. Also, it is important to solve some special situations, i.e. overlapping of patterns.

Klíčová slova

Byznys proces, BPMN, XML, graf, modelovací nástroj, vyhledávání vzorů, BPMN model, analýza, vzory byznys procesů

Keywords

Business process, BPMN, XML, graph, modelling tool, finding patterns, BPMN model, analysis, workflow patterns

Citace

Martin Ludvík: Analýza diagramů byznys procesů, diplomová práce, Brno, FIT VUT v Brně, 2009

Analýza diagramů byznys procesů

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Petra Weisse Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Martin Ludvík
26. května 2009

Poděkování

Děkuji panu Ing. Petrovi Weissovi za vedení a pomoc při tvorbě této diplomové práce. Dále bych chtěl poděkovat své rodině, která mě po celou dobu studia podporovala. A také Julii Kocůnové, která mi byla po většinu studia inspirací do další práce.

Obsah

Kapitola 1

Úvod

1.1 Slovo úvodem

Modelování byznys procesů je v dnešní době velmi rychle rozšiřující se oblast informačních technologií. Stojí totiž na půl cesty mezi managery a pracovníky IT, přičemž počty těchto pracovníků ve firmách rostou velmi rychlým tempem. Obě strany diagramům byznys procesů poměrně dobře rozumí a dokáží se na jejich základě dohodnout poměrně rychle o věcech, které by jinak mohli řešit velice dlouhou dobu. A to například už z toho prostého důvodu, že slovo *proces* je v každé zájmové skupině chápáno částečně jinak, což samo zapříčiňuje jistou komunikační bariéru. V oblasti informačních technologií je proces chápán především jako softwarový proces a ten jako takový je součástí byznys procesu. Kdežto mezi managery je proces chápán především jako samotný byznys proces. Tento je definován jako *posloupnost kroků, která respektuje určitá byznys pravidla a vede k nějakému zisku (hmotnému i nehmotnému).*

Cílem diagramů byznys procesů je zaznamenání jistého postupu při výrobě, vývoji nebo jiné opakované aktivitě ve společnosti (firmě). Tato jejich vlastnost je nesmírně cenná, a to především z důvodu zachování procesů (a to především jejich části, které vedou k zisku) i v případě, že firmu opustí lidé, kteří tyto procesy zaváděli. Také umožňují optimalizaci jednotlivých procesů na základě těchto diagramů.

Další problémy spočívají v nástrojích a postupech, které každé odvětví používá. V oblasti informačních technologií jsou rozšířeny určité modelovací techniky (např. UML), ale v oblasti managementu jsou většinou používány techniky naprosto odlišné. Avšak existuje například postup tranformace z *Business Process Modeling Notation* (dále jen BPMN) do diagramu aktivit. Tím je umožněno převádět BPMN diagramy do formy, která je ještě lépe pochopitelná pro IT odborníky.

Kromě této transformace na diagramy aktivit exituje také možnost převodu do notace Business Process Execution Language for Web Services (dále jen BPEL). A právě pro tento převod je velmi výhodné vyhledávání vzorů v diagramech byznys procesů. Postup vyhledávání bude popsán v dalších kapitolách.

1.2 Členění kapitol

V této části bude stručně nastíněno členění jednotlivých kapitol a ve zkratce bude popsán jejich obsah.

V kapitole 2 je popsána historie, vývoj a současný stav BPMN. Kromě toho je zde

vysvětleno použití BPMN a principy samotné notace.

Kapitola 3 se týká workflow patterns. Vzory jsou zde vyjmenovány a popsány. Vzorům, které je třeba vyhledávat v rámci diplomové práce je věnován větší prostor. Tyto významné vzory jsou také vyobrazeny.

V kapitole 4 jsou popsány základy teorie grafů a je osvětleno pozadí vazby mezi grafy a strukturou XML dokumentů. Je popsán princip vyhledávání vzorů v grafech a různé abstrakce XML souboru a přístupu do nich.

V kapitole 5 jsou nastíněny vlastnosti několika modelovacích nástrojů, které podporují notaci BPMN. Jsou zhodnoceny z hlediska výhod a nevýhod pro daný úkol, tedy především kvality exportu do XML souboru.

V kapitole 6 je popsán způsob uložení vzorů v XML souborech. Je zde uvedena struktura těchto souborů a jsou zde nastíněny možnosti, jak by bylo možné soubory rozšířit o další elementy.

Kapitola 7 je věnována způsobu vyhledávání vzorů v rámci diagramů. Pro správné uvedení do souvislostí je na počátku uveden způsob uložení diagramů, které produkuje zvolený systém z kapitoly 5. Mimo samotného vyhledávání se také zabývá problémy, které při vyhledávání mohou nastat, jako je překryv vzorů, jejich zanoření, atd.

Kapitola 8 poukazuje na specifické rysy implementovaného nástroje pro analýzu diagramů. Jsou uvedeny jeho vlastnosti, omezení a jsou nastíněny možnosti jeho případného rozšíření.

V závěru je učiněno ohlédnutí za celou prací a provedeno její finální zhodnocení. Dále jsou zde vyzdvihnuty nejdůležitější rysy práce s ohledem na další možný rozvoj tématu.

Kapitola 2

Business Process Modeling Notation

V této kapitole je popsána notace BPMN, která je spravována skupinou *Object Management Group* (dále jen OMG).

2.1 Vývoj BPMN

Motivací pro vznik notace byla snaha o zacelení mezery, která byla vytvořena neexistencí grafické notace pro *Business Process Modeling Language* (BPML).

Za vznikem notace BPMN stála iniciativa *Business Process Management Initiative* (BPMI). V roce 2001 vznikla pracovní skupina, která začala notaci vytvářet. Po několika předběžných verzích byla v květnu roku 2004 vydána verze 1.0. V roce 2005 bylo BPMN převzato skupinou OMG, která vydala upravenou verzi 1.0 v únoru roku 2006.

V současné době je aktuální verze 1.1, která byla vydána v únoru roku 2008. Tato verze má několik nedostatků (např. některé součásti nemají přesně definovanou sémantiku) a příliš se neliší od předchozí. Mnou vytvořená práce vychází z této verze specifikace.

O opravu některých záležitostí a rozšíření možností BPMN se pokusí verze 2.0, která je v současné době ve stádiu RFP, neboli request for proposals (výzva k předložení návrhů).

2.2 BPMN a jeho vlastnosti

Jednotlivé součásti BPMN nemají v češtině ustanovenu jednotnou terminologii, a proto v následujícím textu bude používána především původní terminologie v angličtině. V případě použití českého výrazu bude pro zachování konzistence (v místech, kde je to vhodné) uveden i jeho ekvivalent v anglickém jazyce.

Business Process Modeling Notation (BPMN) je standardizovaná grafická reprezentace pro kreslení obchodních procesů ve firemní sféře.

Jako taková se skládá z mnoha grafických prvků. Prvky lze rozčlenit do několika skupin. Těmito skupinami jsou:

1. Flow Objects
2. Connecting Objects
3. Swimlanes

4. Artifacts

V následných podkapitolách je vysvětleno použití a popsány vlastnosti jednotlivých skupin.

2.2.1 Flow Objects

Společně s connecting objects tvoří kostru diagramu, která vystihuje principy procesů. Flow objects jsou opět členěny na několik podtříd:

1. Events (Události)
2. Activities (Aktivity)
3. Gateways (Brány)

Events (Události)

Event (událost) je něco, co se stane v průběhu byznys procesu. Události ovlivňují běh procesu. Většinou mají určitou příčinu a také následek. V BPMN jsou označovány kolečkem. Uvnitř kolečka může být značka, která u událostí odlišuje jednotlivé příčiny nebo následky.

Pokud je okraj kolečka jednoduchý, jedná se o počáteční událost. Dvojitý okraj značí událost v průběhu procesu. Kolečko s tučným okrajem je koncová událost. Názorně je to vidět na obrázku 2.1. Všechny následující obrázky týkající se BPMN diagramů pocházejí z nástroje *Business Process Visual ARCHITECT 2.4*[1]. Popis tohoto a dalších nástrojů je uveden níže (viz kap. 5).

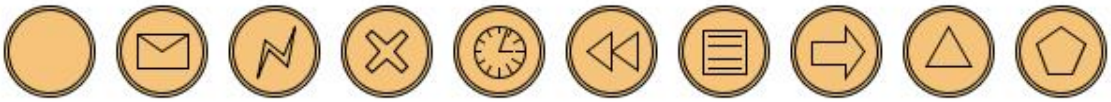
Obrázek 2.1: Ukázka možných druhů událostí (Events)

Počáteční událost neboli „start event“ nám značí, jak proces začíná. Existuje několik druhů počátečních událostí odlišených pomocí obrázku uvnitř kolečka. Tyto specifické počáteční události upřesňují, co způsobí počátek procesu (viz Obrázek 2.2) [2].

Obrázek 2.2: Ukázka možných druhů počátečních událostí (Start Events)

[1] lze stáhnout na WWW adrese http://www.visual-paradigm.com/product/bpva/
[2] Detailní popis jednotlivých druhů událostí lze nalézt v [7]

Události v průběhu procesu neboli „intermediate event“ (viz Obrázek 2.3) se mohou nacházet kdekoliv mezi počáteční a koncovou událostí. Ovlivňují běh procesu, ale nemohou ho odstartovat nebo ukončit.



Obrázek 2.3: Ukázka možných druhů událostí v průběhu procesu (Intermediate Events)

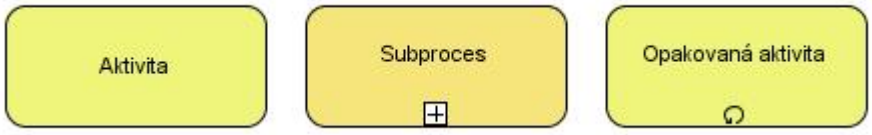
Poslední skupinou událostí jsou koncové události „end event“. Jak jejich jméno napovídá, ukazují, kde daný proces končí. Jako u všech předchozích událostí je možnost odlišit druh koncové události pomocí značky uvnitř kolečka (viz Obrázek 2.4).

Obrázek 2.4: Ukázka možných druhů koncových událostí (End Events)

Activities (Aktivity)

Aktivita je činnost vykonaná během procesu. Může být *atomická* – taková, která nelze dále dělit. Nebo může být *složená* – taková, kterou lze rozdělit na další subprocesy. Atomická aktivita je někdy také nazývána jako *task*.

Aktivita se zakresluje jako obdélník s oblými rohy. Pro případ subprocesu se používá značka plus uvnitř obdélníku. Aktivita může mít definované opakování. Toto je značeno pomocí kulaté šipky (ve tvaru smyčky). Všechny tyto typy lze vidět na obrázku 2.5.



Obrázek 2.5: Ukázka vyobrazení aktivit

Gateways (Brány)

Brány jsou používány pro kontrolu běhu procesu. Umožňují větvení a spojování jednotlivých cest v procesu.

Obrázek 2.6: Ukázka brány (Gateway)

Jsou značeny čtvercem otočeným o 45 stupňů. Obrázek uvnitř čtverce určuje druh chování brány (viz Obrázek 2.6).

Na tomto místě je funkce bran nastíněna jen stručně, protože význam brány se výrazně mění podle jejího konkrétního použití v kontextu diagramu. To je mimo jiné zapříčiněno i tím, že sémantika jednotlivých bran v aktuální verzi notace BPMN není zcela ujasněna.

Jejich funkce by se dala s trochou nadsázky přirovnat k řídícím konstrukcím běžného programovacího jazyka. Lze pomocí nich běh procesu větvit (ať už podmíněně nebo ne), nebo naopak spojovat několik toků v rámci jednoho procesu.

Označení jednotlivých bran pomocí piktogramů uvnitř je vidět na obrázku 2.7. První na obrázku je brána *Exclusive gateway*, která provádí výlučný výběr (Exclusive choice – odtud název brány). Například při větvení běhu procesu vedou z této brány dvě nebo více cest. Proces se pak může vydat dále jen po jedné z nich, což je zajištěno podmínkou, která je přiřazena ke každé z větví. Alternativní notace uvádí jako piktogram uvnitř brány velké X.

Druhá z bran na obrázku je nazývána *Inclusive gateway*. Narozdíl od brány Exclusive jsou i přes splnění jedné z podmínek dále vyhodnoceny i podmínky ostatní. V případě, že je splněna některá z následujicích podmínek, proces se vydá i touto další cestou. Z brány se tedy proces může dál vydat až N cestami, kde N odpovídá počtu cest, u kterých byly splněny podmínky.

Třetí bránou je *Complex gateway*. Tato je používána v situacích, kdy nelze snadno použít jiný druh brány. Lze pomocí ní spojit několik bran do jedné s komplexnější podmínkou. Například lze rozhodovat v rámci jedné brány nejdříve na základě dat v procesu a pak podle stavu příchozí cesty.

Čtvrtá brána je nazývána *Parallel gateway*. Umožňuje paralelní běh několika cest v procesu. Tato brána se většinou používá jak k dělení běhu procesu, tak i k jeho spojování.

Všechny tyto brány lze použít k dělení cest a případně k jejich následujícímu spojování.

Obrázek 2.7: Jednotlivé typy bran

2.2.2 Connecting Objects

Connecting objects (spojovací objekty) propojují jednotlivé flow objects a tím s nimi vytváří základní strukturu procesu. Connecting objects jsou rozděleny do následujících skupin:

1. Sequence Flow (Sekvence)
2. Message Flow (Tok zpráv)
3. Association (Asociace)

Sequence Flow (Sekvence)

Sekvence se používá pro vyjádření pořadí (sekvence), v jakém je proces prováděn. Sekvence musí mít na svém počátku i konci některý z prvků *gate*, *activity* nebo *event*. Posloupnost nesmí překročit hranice entity (vyjádřené např. poolem - viz Sekce 2.2.3) nebo subprocesu.

Sekvence je zobrazována jako nepřerušovaná šipka zakončená vyplněným trojúhelníkem (viz Obrázek 2.8).

Obrázek 2.8: Ukázka sekvence (Sequence Flow)

Message Flow (Tok zpráv)

Je používána pro vyjádření toku zpráv mezi dvěma oddělenými procesy. Tok zpráv může být vázán na pool nebo na aktivitu mimo pool.

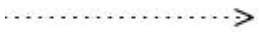
Tok zpráv je zobrazován jako přerušovaná šipka zakončená prázdným trojúhelníkem (viz Obrázek 2.9).

Obrázek 2.9: Ukázka toku zpráv (Message Flow)

Association (Asociace)

Asociace je používána k zobrazení vstupů a výstupů aktivit (např. vstup dat).

Asociace je vyobrazena jako tečkovaná šipka s jednoduchým zakončením (viz Obrázek 2.10).



Obrázek 2.10: Ukázka asociace (Association)

2.2.3 Swimlanes

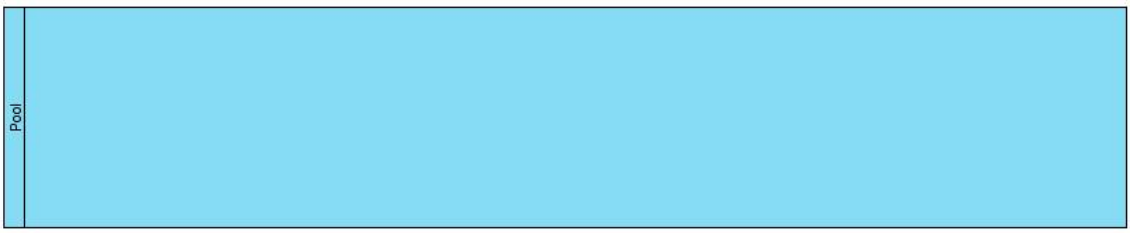
Swimlanes jsou používány pro oddělení rolí v rámci procesu. Swimlanes se dělí na 2 typy, které jsou mezi sebou ovšem vzájemně těsně spjaty. Těmito typy jsou:

1. Pool
2. Lane

Toto dělení je asociací k bazénu (pool) a jeho jednotlivým plaveckým drahám (lane). A stejný je i princip použití pool a lane.

Pool (Bazén)

Pool vymezuje entitu participující na procesu. Grafická podoba je vidět na obrázku 2.11.



Obrázek 2.11: Ukázka Pool

Lane (Dráha)

Lane stejně jako dráha v případě opravdového bazénu vymezuje jen podčást (podentitu) poolu. Dráhy jsou používány pro organizování a kategorizování aktivit. Grafická podoba je vidět na obrázku 2.12.

2.2.4 Artifacts (Artefakty)

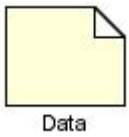
Poslední skupinou prvků v BPMN jsou artifacts (artefakty). Jedná se především o prvky, které se snaží rozšířit vyjadřovací schopnost diagramu a uvést proces do kontextu prostředí, ve kterém běží. Rozlišujeme zde 3 základní skupiny:

1. Data Object (Datový objekt)
2. Group (Skupina)
3. Annotation (Poznámka)

Obrázek 2.12: Ukázka poolu s jednotlivými swimlanes

Data Object (Datový objekt)

Datové objekty jsou mechanismem, kterým lze vyjádřit, jaká data jsou požadována nebo produkována aktivitami. Jsou připojovány k aktivitám pomocí asociací. Datový objekt vypadá například jako na obrázku 2.13.



Obrázek 2.13: Ukázka datového objektu (Data Object)

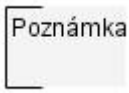
Group (Skupina)

Je použitelná pro dokumentační nebo analytické účely. Je reprezentována obdelníkem se zaoblenými rohy, který je nakreslen čerchovanou čárou (viz Obrázek 2.14). Může být použita například k vyznačení logické souvislosti mezi skupinou aktivit.

Obrázek 2.14: Ukázka skupiny (Group)

Annotation (Poznámka)

Poznámka je způsob, jímž tvůrce diagramu může přidat nějakou textovou informaci pro uživatele diagramu (viz Obrázek 2.15).



Obrázek 2.15: Ukázka poznámky (Annotation)

2.2.5 Shrnutí notace BPMN

Notace je v současné době považována za standard v modelování byznys procesů. Je stále ve fázi úprav a v této práci není popsána zcela vyčerpávajícím způsobem. Její aktuální verzi je možné stáhnout na stránkách www.bpmn.org.

V této části bylo čerpáno především z [7], [10], [9] a [11].

Kapitola 3

Workflow Patterns

Workflow Patterns jsou ustálená řešení typických situací při modelování procesů. Popisují, jak modelovat specifickou návaznost aktivit. Pro znázornění vzorů je v této kapitole použita notace BPMN, jež je výše vysvětlena.

Nalezením a rozdělením těchto vzorů se zabývá *W. van der Aalst* ve své práci *Workflow Patterns* (viz [3]). Tuto práci uvedl do kontextu s diagramy aktivit například *S. A. White* (viz [10]). V obou je popsáno základních 21 vzorů, které jsou rozděleny do šesti skupin:

1. Basic Control Flow Patterns
2. Advanced Branching and Synchronization Patterns
3. Structural Patterns
4. Multiple Instances (MI)
5. State-based Patterns
6. Cancellation Patterns

Vzorům v rámci těchto skupin se věnuje několik následujících podkapitol.

Při dalším výzkumu těchto vzorů byl jejich počet rozšířen na více než 40. Z důvodu rozsáhlosti této problematiky čtenáře odkazuji na dokument [4], ze kterého jsem při tvorbě této práce také čerpal.

3.1 Basic Control Flow Patterns

Tato skupina postihuje základní typy vzorů, které poskytují možnosti toho nejelementárnějšího řízení běhu procesu. Patří sem 5 následujících vzorů:

Sequence – jedna aktivita v procesu je umožněna až po ukončení předcházející ve stejném procesu

Parallel Split – bod v běhu procesu, kde se jedna cesta dělí na více vláken, která mohou být prováděna paralelně. A proto mohou být prováděna v libovolném pořadí nebo simultálně

Synchronization – bod v běhu procesu, kde se sbíhá několik paralelních cest. Zadrží běh procesu, dokud není dokončeno provádění všech předchozích aktivit.

Exclusive Choice – vybere jednu z cest v běhu procesu, na které se proces větví na základě splnění podmínek, které jsou jednotlivým cestám přiřazeny

Simple Merge – místo v běhu procesu, kde se schází více cest běhu procesu. V tomto případě je to bez synchronizace.

3.2 Advanced Branching and Synchronization Patterns

Tato skupina poskytuje pokročilejší možnosti větvení a synchronizace.

Multiple Choice – místo v běhu procesu, kde je vybráno několik cest ze všech alternativ

Synchronizing Merge – slučuje několik přicházejících cest do jedné a to se synchronizací

Multiple Merge – místo, kde se spojují dvě nebo více cest procesu do jedné a to bez synchronizace. Řídící token [1], který sleduje tok procesu, bude vyslán opakovaně pro každý spříchozí token z kterékoliv větve.

Discriminator – místo, kde se čeká na příchod tokenu z kterékoliv příchozí větve. Token je vyslán jen pro první příchozí událost.

N-out-of-M Join – má obdobnou funcki jako Discriminator, ale může čekat na více příchozích událostí

3.3 Structural Patterns

Dva vzory zahrnuté do této skupiny se zabývají smyčkami a nezávislostí jednotlivých cest v procesu.

Arbitrary Cycles – místo, kde může být prováděna jedna nebo více aktivit opakovaně

Implicit Termination – ukončí cestu v procesu, jestliže už není žádná aktivita, která by mohla proběhnout, aniž by byl ukončen celý proces

3.4 Multiple Instances (MI)

Následující čtyři vzory popisují jak jsou tvořeny několikanásobné instance nebo kopie aktivit. V některých situacích mohou být tyto vzory nahrazeny standardními smyčkami.

MI without synchronization – generuje instance jedné aktivity bez následné synchronizace

MI with a prior known design time knowledge – generuje instance jedné aktivity, kdy počet instancí je znám v době návrhu (se synchronizací)

MI with a prior known runtime knowledge – generuje instance jedné aktivity, kdy počet instancí může být rozhodnut za běhu procesu (funguje jako smyčka *FOR*, ale paralelně)

MI without a prior runtime knowledge – generuje instance jedné aktivity, kdy počet instancí nemůže být rozhodnut (funguje jako smyčka *WHILE*, ale paralelně)

[1] Pojem token zde odpovídá přibližně tokenu, jenž se vyskytuje v Petriho sítích. Je to určitý prvek, který sleduje tok procesu.

3.5 State-based Patterns

V následující skupině jsou uvedeny vzory, které popisují chování procesu, pokud jsou tyto procesy ovlivňovány nějakými vnějšími událostmi.

Deferred Choice – provede se jedna z možných alternativních cest. Výběr cesty není založen na datech známých v době provádění, ale spíše na nějaké události (např. rozhodnutí uživatele)

Interleaved Parallel Routing – provede se několik aktivit v libovolném pořadí, ale žádné dvě se neprovedou současně (např. sdílení prostředků)

Milestone – umožňuje provést jistou aktivitu kdykoliv před dosažením milníku, ale již ne po jeho dosažení

3.6 Cancellation Patterns

Následující vzory vyjadřují, jak může dokončení aktivity způsobit zrušení aktivity nebo celé skupiny aktivit.

Cancel Activity – zastaví provádění aktuální aktivity

Cancel Case – zastaví provádění celého procesu

3.7 Vybrané vzory pro vyhledávání

Pro analýzu, která je v rámci diplomové práce prováděna, není třeba vyhledávat všechny vzory. Nyní jsou uvedeny a detailněji popsány vzory (a jejich kombinace), které budou vyhledávány.

3.7.1 Sequence

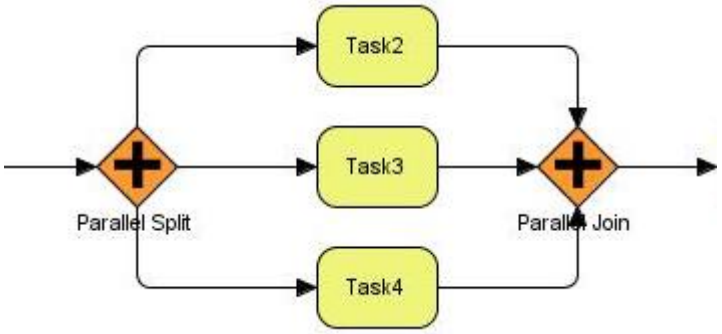
Jako první je uveden tento nejjednodušší vzor, který je definován jako posloupnost aktivit, jež jsou prováděny jedna po druhé. Toto je znázorněno na obrázku 3.1. Na obrázku lze vidět sekvenci aktivit. Místo kterékoliv z nich by se mohla vyskytnout jakákoliv událost (Event) a vzor sekvence by zůstal zachován. Mimo to by mohl být počet aktivit snížen na pouhé dvě, jelikož uspořádání lze definovat na dvou a více prvcích.

3.7.2 Kombinace Parallel Split a Synchronization

Tato kombinace vzorů naznačuje paralelní zpracování, kdy dochází k rozdělení cesty běhu procesu do několika cest a jejich následnému spojení. S tím, že v bodě spojení dochází k čekání na všechny paralelní cesty. Tato kombinace je znázorněna na obrázku 6.5

Opět je možné nahradit některé aktivity (Task2 – Task4) událostmi a zvýšit počty prvků mezi jednotlivými bránami (Gateway).

Obrázek 3.1: Vzor Sequence



Obrázek 3.2: Kombinace Parallel Split a Synchronization

3.7.3 Kombinace Exclusive Choice a Simple Merge

Tato kombinace na svém počátku umožní výběr jedné z cest v pokračování procesu. A na straně druhé očekává příchod tokenu pouze z jedné z cest, které spojuje. Jakmile tento token přijde, tak je okamžitě prováděna další aktivita nacházející se za spojením. Celá tato posloupnost je uvedena na obrázku 3.3

I pro tuto kombinací platí výše uvedená tvrzení o počtu prvků mezi bránami a o záměně aktivit za události.

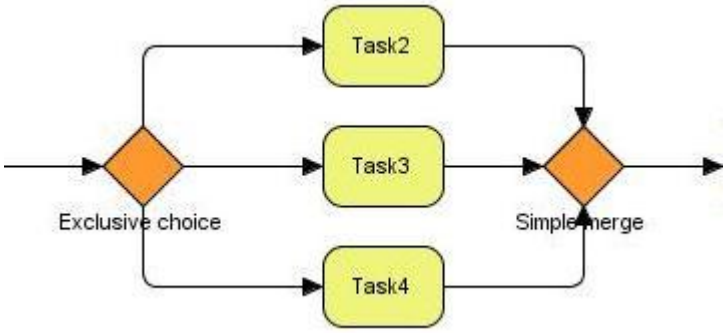
3.7.4 Kombinace Multiple Choice a Multiple Merge

V této kombinaci dochází nejdříve k rozdělení cesty diagramu do několika alternativních cest. Z těchto cest může být zvolena jedna až N, kde N je počet alternativ. Na konci této kombinace dochází opět ke spojování cest do jedné. Avšak za každou cestu, z které dorazil řídící token, je provedena aktivita, jež se nachází za bránou, kde se cesty sbíhají (viz Obrázek 3.4).

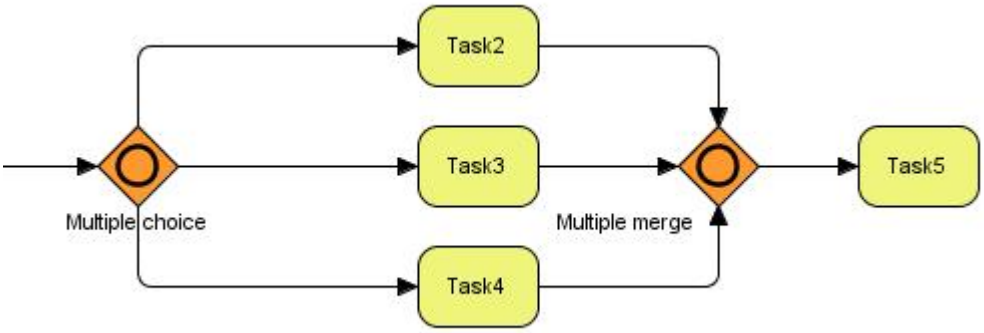
Opět lze uplatnit výše uvedené tvrzení o událostech a počtu mezikroků v oddělených cestách.

3.7.5 Cykly

Posledními vyhledávanými konstrukcemi v diagramech jsou cykly. A to cykly dvou typů:



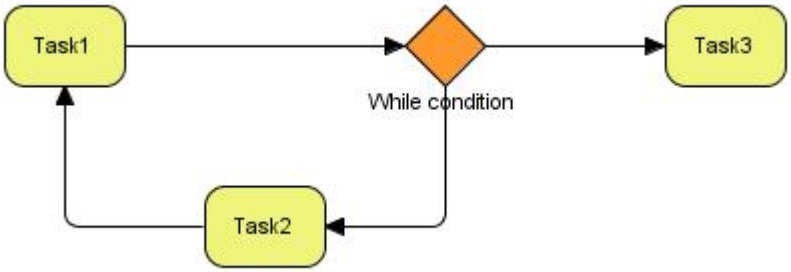
Obrázek 3.3: Kombinace Exclusive Choice a Simple Merge



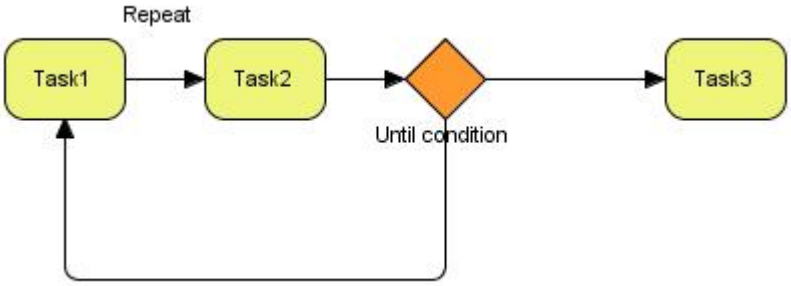
Obrázek 3.4: Kombinace Multiple Choice a Multiple Merge

1. Cyklus typu *WHILE* – nemusí se provést ani jednou
2. Cyklus typu *REPEAT UNTIL* – provede se aspoň jednou

V obou případech se může cyklicky provádět více než jedna aktivita, případně událost. Cyklus typu *WHILE* je na obrázku 3.5 a cyklus typu *REPEAT UNTIL* je na obrázku 3.6



Obrázek 3.5: Cyklus typu WHILE



Obrázek 3.6: Cyklus typu REPEAT UNTIL

3.8 Shrnutí

V této kapitole byly popsány základní workflow patterns. Kromě nastínění funkce všech vzorů byly určeny vzory, které jsou v diplomové práci vyhledávány. Tyto byly popsány detailněji a byla doplňena jejich grafická reprezentace.

Kapitola 4

Grafy a jejich vazba na XML

Vzhledem k tomu, že BPMN notace je grafická, je více než vhodné převést tuto notaci do nějaké textové reprezentace, která se bude lépe zpracovávat. V této práci je za tuto formu zvolen *Extensible Markup Language* (dále jen XML), protože některé nástroje pro modelování byznys procesů dokáží exportovat diagramy do XML souborů (viz kap. 5).

V této kapitole je velmi stručně popsáno několik definic z teorie grafů a vazba mezi grafy a XML. Dále jsou nastíněny vlastnosti a historie XML a jednotlivé přístupy k procházení XML stromů (jako např. DOM).

4.1 Grafy

Teorie grafů je odvětví matematiky, které je sice velice formální, ale produkty této matematické disciplíny využívají často i lidé, jenž si to vůbec neuvědomují.

V následující části jsou definovány 2 základní typy grafů – *neorientované* a *orientované*.

Definice 1: *Neorientovaný graf* je trojice $G = (V, E, \epsilon)$ tvořená neprázdnou konečnou množinou V, jejíž prvky nazýváme *vrcholy*, konečnou množinou E, jejíž prvky nazýváme *neorientovanými hranami*, a zobrazením ϵ, které nazýváme *vztahem incidence* a které každé hraně $e \in E$ přiřazuje jedno- nebo dvouprvkovou množinu vrcholů.

U takovýchto grafů nezáleží na orientaci hran, narozdíl od grafů definovaných v následující podkapitole.

Definice 2: *Orientovaný graf* je trojice $G = (V, E, \epsilon)$ tvořená neprázdnou konečnou množinou V, jejíž prvky nazýváme *vrcholy*, konečnou množinou E, jejíž prvky nazýváme *orientovanými hranami*, a zobrazením $\epsilon : E \to V^2$, které nazýváme *vztahem incidence*. Toto zobrazení přiřazuje každé hraně $e \in E$ uspořádanou dvojici vrcholů (x,y). Prvý z nich, x, nazýváme *počátečním vrcholem hrany* a značíme jej *PV(e)*. Druhý nazýváme koncovým vrcholem hrany a značíme jej *KV(e)*.

Definice 3: *Podgraf* – Graf H je podgraf grafu G, jestliže

$$V(H) \subseteq V(G) \text{ a } E(H) \subseteq E(G) \cap \binom{V(H)}{2},$$

Jinými slovy, podgraf vznikne vymazáním některých vrcholů původního grafu, všech hran do těchto vrcholů zasahujících a případně některých dalších hran.

Definice 4: *Izomorfismus* – Grafy G, G' se nazývají vzájemně *izomorfní*, když existují dvě vzájemně jednoznačná zobrazení $f : V \to V'$ a $g : E \to E'$ taková, že zachovají vztahy incidence ϵ a ϵ'. Přesněji, když pro každou hranu $e \in E$ platí:

$$\epsilon(e) = (x, y) \Leftrightarrow \epsilon'(g(e)) = (f(x), f(y)),$$

popř.

$$\epsilon(e) = \{x, y\} \Leftrightarrow \epsilon'(g(e)) = \{f(x), f(y)\}$$

v závislosti na tom, zda se jedná o grafy orientované nebo neorientované.
Vztah izomorfismu značíme $G \cong G'$.

Vzor v grafu je podgraf, který je izomorfní s jiným grafem, který je předlohou pro daný podgraf. Vyhledávání vzorů je tedy určování izomorfismu mezi jednotlivými podgrafy grafu G a vzorového grafu H.

Definice 5: *Kořenový strom* je orientovaný graf, v němž existuje význačný vrchol r, tzv. *kořen*, takový, že do kořene nevede žádná hrana, do každého jiného vrcholu vede přesně jedna hrana a navíc jsou všechny vrcholy z kořene r orientovaně dostupné.

Kořenový strom patří k nejužitečnějším a nejpoužívanějším grafům vůbec. Jeho definice zde je uvedena jako ukázka přímé návaznosti mezi grafy a XML. Samotný kořenový strom může být dále specializován některými vlastnostmi a může tak vzniknout například: *Binární kořenový strom*, *Vyvážený strom*, atd. Tyto speciální stromy nejsou však pro účely této práce tolik podstatné a jsou zde zmíněny jen pro doplnění přehledu. Více se lze dozvědět v knize *Grafy a jejich aplikace* [5], ze které bylo v celé této kapitole čerpáno. Především jsou z ní převzaty definice v celé části 4.1.

Model kořenového stromu pak přímo odpovídá interpretaci modelu XML.

4.2 Extensible Markup Language

Extensible Markup Language vychází z výše zmiňovaného *kořenového stromu*. Je to obecný značkovací jazyk, který je v současné době využíván ve většině případů, kdy je nutné uložit nějakou strukturovanou textovou informaci (např. konfiguraci programu, komunikaci přes počítačovou síť, atd.). Postup, při kterém je využíváno jedno řešení, které je otestováno a plně funkční, je v mnoha ohledech výhodnější než používání různých proprietárních řešení jednotlivých tvůrců.

4.2.1 Vývoj XML

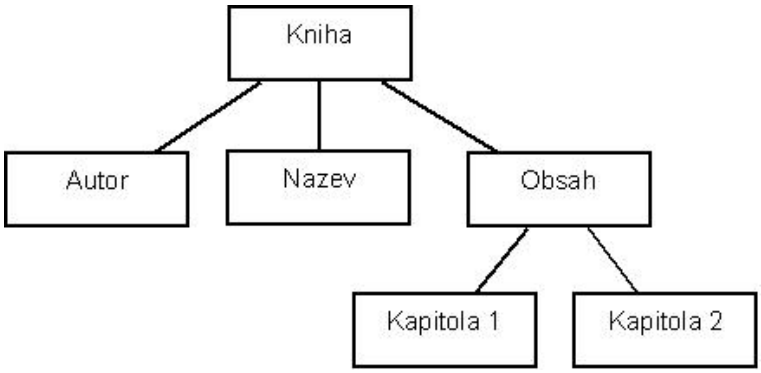
XML vzniklo jako nástupce *Standard Generalized Markup Language* (SGML). Samotné SGML bylo přijato jako standard ISO v roce 1986. Jeho největším úspěchem byl *HyperText Markup Language* (HTML). Tento se prosadil především jako jazyk pro zápis webových dokumentů.

Vývoj XML začal v roce 1996 a stal se doporučením W3C od ledna 1998. Od té doby XML prošlo několika verzemi, z nichž je v současné době (prosinec 2008) aktuální verze XML 1.1. O vývoj a udržování standardu se zasazuje organizace *World Wide Web Consortium* (W3C) [1]. Aplikací XML je například *Extensible HyperText Markup Language* (XHTML), *Simple Object Access Protocol* SOAP, atd.

[1] viz http://www.w3.org/

4.2.2 Struktura XML

Struktura XML v modelu je dána uzly, které jsou uspořádány v kořenovém stromě (viz Obrázek 4.1). XML se skládá ze základních součástí, které jsou nazývány *elementy*, *tagy*, *entity* a *atributy*.



Obrázek 4.1: Ukázka XML stromu

Vlastní informace jsou do XML předávány dvěma způdoby a to pomocí obsahu *elementu* nebo přes *atributy*.

Aby byl dokument považován za správně strukturovaný, musí mít nejméně následující vlastnosti (viz [8]):

1. Musí mít právě jeden kořenový element.
2. Neprázdné elementy musí být ohraničeny startovací a ukončovací značkou. Prázdné elementy mohou být označeny tagem „prázdný element“.
3. Všechny hodnoty atributů musí být uzavřeny v uvozovkách. A to jednoduchých nebo dvojitých, ale jednoduchá uvozovka musí být uzavřena jednoduchou a dvojitá dvojitou. Opačný pár uvozovek může být použit uvnitř hodnot.
4. Elementy mohou být vnořeny, ale nemohou se překrývat; to znamená, že každý (ne kořenový) element musí být celý obsažen v jiném elementu.

Níže (viz Obrázek 4.1) je znázorněna ukázka části XML dokumentu v textové podobě.

```
<?xml version="1.0" encoding="UTF-8"?>
<kniha>
        <autor>John Doe</autor>
        <nazev>Nazev knihy</nazev>
        <obsah>
                <kapitola>Kapitola 1 - Uvod</kapitola>
                <kapitola>Kapitola 2 - Zaver</kapitola>
        </obsah>
</kniha>
```

Ukázka XML 4.1: Ukázka struktury XML

4.2.3 Reprezentace XML

Pro přístup k jednotlivým uzlům ve stromě je používáno několik metod. Nejčastěji jsou používány dvě. A těmi jsou *Document Object Model* (DOM) a *Simple API for XML* (SAX). Čerpáno bylo z [6].

Zbytek této části je věnován popisu rozdílů mezi těmito dvěma přístupy

Document Object Model

DOM je to objektově orientovaná reprezentace XML, která v paměti počítače (případně jiného zařízení) vystaví celý strom, který vyjadřuje strukturu XML souboru. Umožňuje číst a modifikovat jednotlivé uzly stromu. Jeho hlavní nevýhodou je právě nutnost udržet v paměti celý strom, což je na pamět poměrně náročné a v případě, že nedochází k náhodnému a častému přístupu k uzlům, zbytečné.

DOM model byl ovšem původně primárně vyvinut pro zpracování HTML elementů pomocí skriptovacího jazyka *JavaScript* na straně klienta.

Simple API for XML

Narozdíl od DOM modelu, SAX nevytváří v paměti celý strom, ale přistupuje ke XML souboru sekvenčně. Dokument se rozdělí na jednotlivé části a postupně se pak volají jednotlivé události ohlašující nalezení konkrétní součásti. V reakci na tuto událost je prováděna činnost, kterou může programátor definovat.

4.2.4 Aplikace XML spojené s BPMN

Na tomto místě jsou zmíněny dvě aplikace jazyka XML, které mají přímou souvislost s BPMN diagramy. A to především z pohledu možnosti exportu BPMN diagramů do jejich XML reprezentace.

XML Process Definition Language

XML Process Definition Language (XPDL) je aplikace XML standardizovaná skupinou *Workflow Management Coalition* (WfMC), která umožňuje přenášení byznys procesů mezi jednotlivými modelovacími nástroji. Umožňuje přenést grafickou reprezentaci i sémantiku jednotlivých byznys procesů. Aktuálně je považován za nejlepší způsob přenosu byznys procesu (více viz [1]).

XML Metadata Interchange

XML Metadata Interchange (XMI) je standard pro výměnu informací ve formě metadat založený na XML, který se nejčastěji používá pro ukládání modelů UML. Může být však použit pro jakákoliv metadata, která popisují metamodel, jenž může být popsán pomocí Meta-Object Facility (MOF).

4.3 Shrnutí

V této kapitole byla popsána souvislost mezi teorií grafů, XML a vyhledáváním vzorů v grafech. Tato záležitost je významná především z pohledu následné analýzy XML souborů exportovaných zvoleným modelovacím nástrojem. Analýza bude založena na vyhledávání

vzorů v XML (přeneseně tedy grafu). A proto jsou zde popsány i principy přístupu ke struktuře XML a některé aplikace XML, které mají souvislost s BPMN.

Kapitola 5

Nástroje pro modelování BPMN

Pro analýzu BPMN diagramů je třeba si zvolit nástroj, který bude později pro jejich tvorbu používán. A to ze dvou důvodů. Prvním z důvodů je ten, že každý z nástrojů může implementovat jen určitou podmnožinu notace BPMN. Druhým je již výše zmiňovaný postup při zpracování diagramů, při kterém bude docházet k analýze XML souborů, do nichž bude diagram exportován. Na těchto a dalších bodech jsou postavena kritéria hodnocení jednotlivých produktů, která jsou uvedena níže.

Pro porovnání byly zvoleny 3 následující nástroje: *Business Process Visual ARCHITECT 2.4* (dále jen BPVA), *BizAgi BPMN Process Modeler* (dále jen BizAgi) a *MagicDraw*. Všechny tyto nástroje jsou poskytovány freeware nebo mají zkušební dobu, po kterou jsou poskytovány zdarma. Seznam dalších nástrojů, které podporují normu BPMN lze nalézt na http://www.bpmn.org/BPMN_Supporters.htm.

5.1 Kritéria hodnocení

Jako kritéria byla zvolena: *bezplatná dostupnost*, *kvalita exportu XML*, *velikost množiny podporovaných prvků notace BPMN* a *multiplatformnost*.

Všechny produkty zvolené pro testování jsou dostupné alespoň po testovací období zdarma nebo například ve verzi licence – community edition.

Dalším zkoumaným atributem je kvalitní export do formátu XML, jelikož na základě něj bude probíhat analýza. XML bylo zvoleno především, protože jde o otevřený formát a lze do něj provádět změny, které se pak zpětně mohou promítnout do diagramu. Například v případě objevení některého ze vzorů, které jsou uvedeny výše (viz Sekce 3.7), je třeba tento úsek diagramu označit a následně k němu přidat popisek, který určí, o jaký vzor se jedná.

V exportu diagramů se jednotlivé nástroje nejvíce liší. Nástroj BizAgi umožňuje export do formátu XPDL, který je zmíněn výše (viz Sekce 4.2.4). Modelovací nástroj MagicDraw používá export do formátu XMI (viz Sekce 4.2.4). Tento je však spíše vhodný pro jednotlivé modely z notace UML. BPVA oproti tomu používá svou vlastní strukturu XML.

Z pohledu podporované podmnožiny prvků notace BPMN jsou všechny nástroje vyrovnané a výrobci také uvádějí stoprocentní kompatibilitu s BPMN ve verzi 1.1.

Multiplatformnost je problémem jen pro nástroj BizAgi, jelikož ten je funkční pouze pod některými verzemi operačního systému Windows a navíc je třeba mít nainstalován Microsoft .NET 2.0 framework. Zbývající dva nástroje jsou založeny na programovacím jazyce JAVA, a proto jsou také dobře přenositelné mezi různými operačními systémy.

Všechny tyto vlastnosti jsou shrnuty v následující tabulce (viz 5.1).

Nástroj	Dostupnost	Export	Platforma	Podmnožina BPMN
BizAgi	Evaluation copy	XPDL	Windows	BPMN 1.1
MagicDraw	Evaluation copy	XMI	multiplatformní	BPMN 1.1
BPVA 2.4	Comunity licence	XML	multiplatformní	BPMN 1.1

Obrázek 5.1: Shrnutí vlastností nástrojů

5.2 Zvolený nástroj

Zvoleným nástrojem se stal *Business Process Visual ARCHITECT 2.4*. A to především díky jeho multiplatformnosti a vlastnímu otevřenému formátu XML exportu, protože při úpravách některého z formátu XMI nebo XPDL by mohlo dojít k narušení kompatibility pro import do jiných nástrojů. I když je totiž XPDL považován za nejlepší způsob uložení byznys procesů, tak stále není schopen vyjádřit všechny vzory, které byly v rámci diagramů byznys procesů identifikovány. Vylepšení XPDL pro vyjádření těchto vzorů je dosaženo pomocí různých rozšíření, jenž jsou implementovány přímo jednotlivými výrobci software a dochází tak ke ztrátě kompatibility, která je hlavní výhodou XPDL.

Takto je sice zvoleno řešení vázané na určitý nástroj, ale ten je volně dostupný a multiplatformní, takže by neměl být problém tento nástroj zprovoznit na většině operačních systémů.

Výhodou je to, že lze do analyzovaného diagramu vložit potřebné poznámky získané analýzou a diagram v nástroji opět otevřít.

5.3 Shrnutí

V této kapitole je uvedeno a srovnáno několik nástrojů používaných pro modelování diagramů byznys procesů. Byl zde také vybrán nástroj, který je používán při tvorbě a zobrazení diagramů používaných při analýze v rámci diplomové práce.

Kapitola 6

Uložení vzorů

Ve výše uvedených kapitolách byly uvedeny vzory, které jsou vyhledávány a také je zmíněn nástroj, jenž je používán pro tvorbu diagramů a jehož výstupy jsou následně zpracovávány.

V této kapitole je nahlédnuto na vzory z poněkud jiného úhlu pohledu. Nejdříve je vysvětleno, co je za vzor považováno především po stránce struktury. Následuje obecnější popis způsobu, jakým jsou vzory ukládány.

V dalších částech následuje tedy popis XML struktury, která mnou byla navržena pro ukládání vyhledávaných vzorů.

6.1 Struktura vzoru

Struktura vzoru vychází především z teorie grafů (viz Kapitola 4.1). Je třeba tedy adekvátně postihnout množinu uzlů a množinu hran. Uzly jsou ve vzoru jednotlivé prvky, které jsou uvedeny v části 2.2.1 a jsou vyhledávány. Hrany ve vzorech byly nahrazeny pomocí propojovacích prvků z části 2.2.2 (konkrétně jen pomocí prvku Sekvence). Na ukázkách v odkazované části lze vidět, že je v případě propojovacích prvků důležitý i směr vazby, takže jde o orientované grafy.

Situace je ovšem komplikována skutečností, že narozdíl od přesně definovaných grafů se ve vzorech nachází jistá míra neurčitosti. Ta je dána tím, že vzor nedefinuje například kolik uzlů se za sebou může nacházet nebo do kolika toků řízení se může cesta diagramem větvit. A proto jsem jako řešení tohoto problému zvolil určité *zástupné entity*.

6.1.1 Zástupné entity

Zástupné entity jsou takové části vzoru, které nahrazují nějakou (většinou z více uzlů a hran složenou) část vzoru, která není podstatná pro určení, zda se o daný vzor jedná nebo ne.

Takovouto zástupnou entitou je například sekvence několika aktivit za sebou, které přímo navazují jedna na druhou. V tomto případě nemusí být v některých vzorech důležitý počet těchto aktivit, ale pouze to, na jaký uzel se váže první a poslední aktivita. A také to, že se cesta mezi těmito aktivitami nevětví.

Pokud tedy vytváříme šablonu vzoru, nemůžeme vědět (např. u vzoru Sequence) kolik objektů je do konkrétní realizace vzoru zapojeno. Tento neurčitý počet bývá vyjádřen právě *zástupnou entitou*.

Nyní je uveden seznam těchto zástupných entit a vysvětlena funkce každé z nich:

- Sekvence několika uzlů za sebou
- Rozvětvení s nespecifikovaným počtem větví
- Rozvětvení s nespecifikovaným počtem větví, z nichž každá je složena ze sekvence uzlů

Sekvence několika uzlů za sebou: Je seskupení aktivit (task) a událostí (event), které na sebe lineárně navazují. Pro samotný vzor by nemělo být podstatné, jaké aktivity nebo události se v sekvenci nacházejí. Může se zdát, že tato entita přímo odpovídá jednomu ze vzorů (vzor Sequence), ale není tomu tak, jelikož je používána jako zjednodušení pro vzor a ne přímo v diagramu. Ukázku takové sekvence lze vidět například na obrázku 6.1.

Obrázek 6.1: Ukázka sekvence uzlů

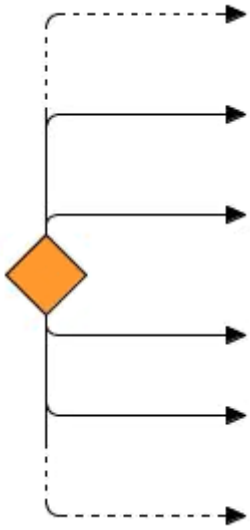
Rozvětvení s nespecifikovaným počtem větví: Je takové místo ve vzoru, kdy dochází k rozvětvení toku řízení ve vzoru a to do nespecifikovaného počtu větví. Pro samotný vzor by nemělo být podstatné, kolik větví se v daném místě nachází, ale je důležité, aby větve obsahovaly stejné posloupnosti aktivit a událostí. Ukázka takového vzoru je na obrázku 6.2.

Rozvětvení s nespecifikovaným počtem větví, z nichž každá je složena ze sekvence uzlů: Je vlastně kombinací dvou předchozích zástupných entit. Specialitou této entity je možnost akceptování dalšího větvení mimo brány v průběhu cest. Je možné si ji prohlédnout na obrázku 6.3.

Tyto zástupné entity pomáhají vystihnout místa ve vzorech, která není dost dobře možné zachytit nějakým jiným, jednodušším způsobem.

6.2 Způsob uložení

V této části je vysvětleno jak jsou uloženy jednotlivé vzory, aby mohly být následně vyhledávány v diagramech. Za nejvhodnější způsob ukládání byl zvolen formát *XML*. A to především z důvodu, že jej lze snadno zpracovávat. Je možné si navrhnout vlastní strukturu, jež odpovídá přesně potřebám aplikace. A v neposlední řadě je tento formát dobře čitelný i pro člověka. Poučený uživatel tak může bez větších problémů přímo ručně editovat uložené vzory.



Obrázek 6.2: Ukázka větvení s nespecifikovaným počtem větví

Kořenový XML element, jenž se nachází v každém vzoru, je označen `pattern` a má vždy atribut `name`, v kterém je specifikován název konkrétního vzoru. Může vypadat tedy například takto (viz 6.1):

```
<pattern name="nazev">...</pattern>
```

Ukázka XML 6.1: Ukázka kořenového elementu

Mezi počátečním a koncovým tagem kořenového elementu musí být popsán celý vzor a jeho struktura.

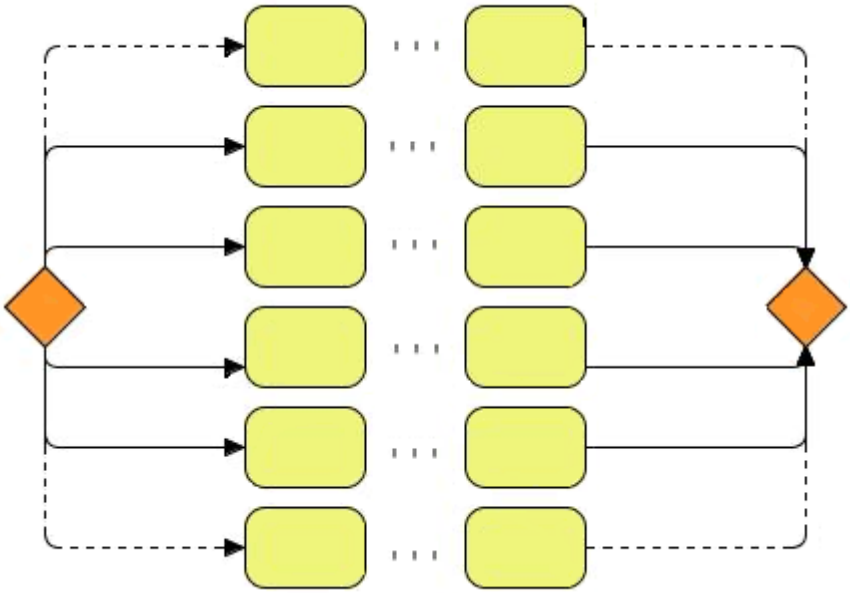
Pořadí uzlů[1] a propojení mezi jednotlivými uzly ve vzoru je řešeno pomocí unikátních identifikátorů pro každý uzel. Propojené prvky[2] obsahují následně ve svých atributech odkazy na své předchůdce nebo následovníky.

Obecně je pro všechny prvky podobná struktura elementu[3]. Tato struktura vypadá následovně (viz 6.2):

[1] Uzlem je v tomto případě myšlen uzel v grafu, který odpovídá konkrétnímu prvku ve vzoru nebo diagramu.

[2] Prvek v tomto případě odpovídá pojmu *uzel*. Tyto pojmy mohou být nadále zaměňovány ve stejném významu.

[3] Element odpovídá elementu v XML.



Obrázek 6.3: Větvení na sekvence uzlů

```
<nazev_prvku id="identifikator" name="nazev" type="typ"
input0="odkaz_na_id_prvniho_predchoziho" ...
inputN="odkaz_na_id_nteho_predchoziho"
   output0="odkaz_na_id_prvniho_nasledneho"
... outputN="odkaz_na_id_nteho_nasledneho" />
```

Ukázka XML 6.2: Ukázka struktury atributů elementu

Význam jednotlivých atributů je takovýto:

id – Je jednoznačný identifikátor elementu. Obsahuje textový řetězec odpovídající řetězci 128 bitů.

name – Je specifický název prvku ve vzoru. Množina jmen se liší podle typu elementu, v němž se atribut nachází. Více o atributu lze nalézt v části 6.3.

type – Je specifický typ prvku ve vzoru. Typy, které se dají použít, se liší podle druhu elementu, v němž se atribut nachází. Více o atributu lze nalézt v části 6.3.

výčet atributů input – Jde o výčet atributů, které obsahují identifikátory (hodnoty atributů id) prvků vzoru, na něž se prvek váže.

výčet atributů output – Jde o výčet atributů, jež obsahují identifikátory (hodnoty atributů id) prvků vzoru, které se vážou na samotný prvek.

Název elementu se liší pro různé uzly ve vzoru. Seznam těchto prvků a specifické hodnoty atributů pro každý z nich, je uveden v další části. Je zde popsán i konkrétní význam jednotlivých hodnot atributů.

6.3 Uložení jednotlivých uzlů a hran

V této části jsou popsány jednotlivé elementy, které se mohou nacházet v XML dokumentu, jenž popisuje konkrétní vyhledávaný vzor.

Jsou uvedeny jednotlivé elementy společně s popisem vazby ke konkrétnímu prvku používaném ve vzoru a s výpisem množiny hodnot některých specifických atributů, a to včetně významu hodnot těchto atributů. Všechny tyto elementy mohou mít atributy, jež jsou uvedeny výše v části 6.2.

6.3.1 Aktivita (Task)

Aktivita je uzel ve vzoru, který odpovídá aktivitě uvedené v části 2.2.1. Element korespondující s tímto uzlem má jméno `task`. Specifická hodnota atributu `name` je `Task` spojená pomlčkou s číslem určujícím pořadí přidání uzlu do vzoru. Atribut `type` může nabývat hodnoty `None`, což značí, že nezáleží na typu aktivity (viz 6.3). Atribut `type` je zde tedy uveden pro případ dalšího rozšiřování systému.

```
<task id="fc423d50-1033-4e42-aa27-64651b0e1079" name="Task-0"
   type="None" />
```

Ukázka XML 6.3: Ukázka struktury elementu Task

6.3.2 Událost (Event)

Událost odpovídá pojmu událost v části 2.1. Element má v tomto případě jméno `event` a hodnota atributu `name` může mít pouze hodnotu `Event` v kombinaci s pomlčkou a pořadovým číslem prvku. Atribut `type` musí také nabývat hodnoty `None` (viz 6.4).

```
<event id="166b62b1-c73e-435d-9e24-fa3985d35a06"
   name="Event-0" type="None" />
```

Ukázka XML 6.4: Ukázka struktury elementu Event

6.3.3 Brány (Gateways)

Brány jsou místa, kde se mohou ve vzoru rozdělovat nebo spojovat toky řízení a odpovídají bránám z části 2.2.1. Element má jméno `gate`. Ale v tomto případě záleží na hodnotě atributu `name` a `type`. Ty mohou podle typu brány ve vzoru mít hodnoty podle tabulky na 6.4. K hodnotě atributu `name` se opět připojuje pomlčka a za ni pořadové číslo prvku. Uložení brány je vidět na ukázce 6.5.

Brána	Hodnota `name`	Hodnota `type`
Exclusive Gateway	ExclusiveGateway	XOR
Inclusive Gateway	InclusiveGateway	OR
Parallel Gateway	ParallelGateway	AND
Complex Gateway	ComplexGateway	Complex

Obrázek 6.4: Kombinace atributu name a type podle typu brány

```
<gate id="d7fa8878-a029-4303-9e47-06291eb491ed"
name="ParallelGateway-0" type="AND" />
```

Ukázka XML 6.5: Ukázka struktury elementu pro Parallel gateway

6.3.4 Sekvence (Sequence Flow)

Sekvence, sloužící pro navazování ostatních prvků ve vzoru, odpovídá sekvenci z části 2.2.2. Entita zastupující tuto sekvenci má název `seqflow`. Atribut `name` může mít hodnotu `SequenceFlow` v kombinaci s pomlčkou a pořadovým číslem prvku. Atribut `type` může nabývat pouze hodnoty `None` (6.6).

```
<seqflow id="69233ea3-f0f4-4b42-a325-e7e8129a62fa"
name="SequenceFlow-2" type="None"
   input0="d7fa8878-a029-4303-9e47-06291eb491ed"
output0="803ec19c-3fae-4d53-90a8-87effece0558" />
```

Ukázka XML 6.6: Ukázka elementu pro Sequence Flow

6.3.5 Sekvence uzlů

Sekvence uzlů je jednou ze zástupných entit popsaných v části 6.1.1. Entita má název `nseq`. Atribut `name` má hodnotu `Nsequence` v kombinaci s pomlčkou a pořadovým číslem prvku. Atribut `type` může nabývat pouze hodnoty `None`. XML zápis pro tuto zástupnou entitu je na ukázce 6.7.

```
<nseq id="bc0c0e7d-bd27-4515-9e2e-4c6dcd55964b"
   name="Nsequence-2" type="None"
   input0="0fac5424-f727-40f8-a8b7-fd3aa444576d"
   output0="25a0ea1c-3480-446a-aff4-b9a668a9cab6" />
```

Ukázka XML 6.7: Ukázka elementu pro sekvenci uzlů

6.3.6 N – násobné větvení

N – násobné větvení je zástupná entita popsaná v části 6.1.1. Entita má název `nbranch`. Atribut `name` má hodnotu `Nbranches` v kombinaci s pomlčkou a pořadovým číslem prvku. Atribut `type` může nabývat pouze hodnoty `None`. Zápis v XML pro tento element může vypadat jako na ukázce 6.8.

```
<nbranch id="b563ba32-ab88-44a8-971c-d5d883536b56"
   name="Nbranches-2" type="None"
   input0="63507913-3462-43dd-95dc-e86d5cfcc368"
   output0="d68a4a37-cc4f-4958-aa28-5c463e266214" />
```

Ukázka XML 6.8: Ukázka zástupného elementu pro vícenásobné větvení

6.3.7 Kombinace větvení a sekvence uzlů

Kombinace větvení a sekvence uzlů je kombinace výše uvedených prvků a je popsána v části 6.1.1. I když je to z logického pohledu kombinace dvou prvků, tak pro tuto situaci existuje vlastní entita, která se jmenuje `matrix`. Atribut `name` nabývá hodnoty `NxN` a atribut `type` nabývá hodnoty `None`. Možný vzhled XML pro tento kombinovaný element je nastíněn na ukázce 6.9.

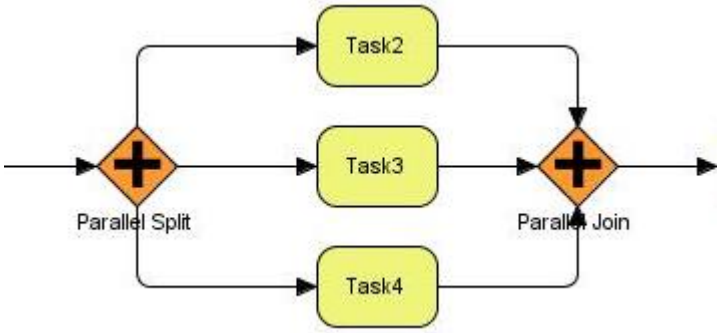
```
<matrix id="35f3f7ac-39a0-47ef-8ac0-a9cd74dffa31"
   name="NxN-2" type="None"
   input0="96f56533-8fc3-42ff-8d9d-bcf324c279ec"
   output0="e1a62cb8-83c1-45c7-87e1-521a2bdecdcc" />
```

Ukázka XML 6.9: Ukázka zástupného elementu pro matici vazeb

6.4 Ukázka struktury souboru se vzorem

Na následujících řádcích je popsána struktura XML souborů. Narozdíl od předchozích částí je zde uveden příklad struktury konkrétního vzoru (viz Obrázek 6.5), jehož vyobrazení je uvedeno ještě před samotným vzorem. Za ukázkou XML (viz 6.10) je uveden popis upřesňující , jak koresponduje obrázek s ukázkou XML.

Konkrétně jde o vzor kombinující Parallel Split a Synchronization. Tento vzor je detailně popsán v části 3.7.2.



Obrázek 6.5: Kombinace Parallel Split a Synchronization

```
<?xml version="1.0" encoding="UTF-8"?>
<pattern name="Parallel Split a Synchronization">
  <gate id="cb2817d1-8a62-4fd2-8cdc-4f2907caff73"
     name="ParallelGateway-0" type="AND"
     output0="1a430e8e-50df-4a73-b3fd-c4f71fd2f64b" />
  <gate id="905cdb85-caf0-415a-8547-b02644fde3cd"
     name="ParallelGateway-1" type="AND"
     input0="1a430e8e-50df-4a73-b3fd-c4f71fd2f64b" />
  <matrix id="1a430e8e-50df-4a73-b3fd-c4f71fd2f64b"
     name="NxN-2" type="None"
     input0="cb2817d1-8a62-4fd2-8cdc-4f2907caff73"
     output0="905cdb85-caf0-415a-8547-b02644fde3cd" />
</pattern>
```

Ukázka XML 6.10: Kombinace Parallel Split a Synchronization v XML předpisu

Přímou korespondenci mezi vzorem a XML předpisem lze vidět na obrázku 6.5 a ukázce XML 6.10. Element s atributem `name=ParallelGateway-0` odpovídá bráně, která je na obrázku označena jako *Parallel Split*. Element s atributem `name=ParallelGateway-1` odpovídá bráně, která je na obrázku označena jako *Parallel Join*.

Spojení těchto bran zajišťuje element `matrix`, ve kterém je na straně vstupu odkaz na první bránu a na straně výstupů odkaz na druhou bránu. Tento element je použit především z toho důvodu, že vzor, který je předlohou, dovoluje nespecifikovaný počet větví vycházejících z první brány a vstupujících do druhé brány. Mezi bránami také může být neomezený počet aktivit nebo událostí a jelikož nás tyto nezajímají při hledání tohoto vzoru, tak si můžeme dovolit je zaznamenat pomocí zástupné entity.

6.5 Shrnutí

V této kapitole byly popsány vzory z pohledu možnosti jejich uložení a struktury jednotlivých elementů, jež se mohou ve vzoru nacházet. Tato struktura byla navržena v poměrně jednoduchém tvaru, aby ji bylo možné v případě dalšího zpracování jednoduše rozšířit a upravit.

Možnost rozšíření do struktury vnáší například atribut `type`, který je využíván pouze u bran. Další možností rozšíření by bylo převedení dalších prvků z BPMN notace (viz 2.2). Toho by šlo docílit vytvořením nového elementu, který by ale splňoval veškeré podmínky, jež jsou uvedeny v části 6.2.

Kapitola 7

Vyhledávání vzorů v diagramu

V této kapitole je popsán postup, který je používán pro vyhledávání vzorů v diagramu.

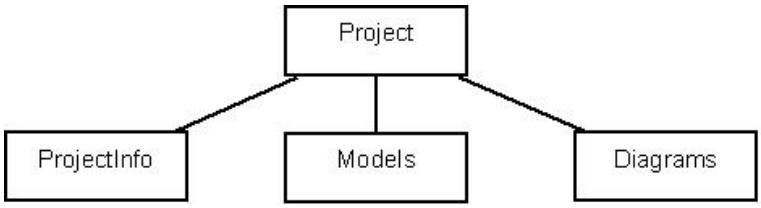
Před vlastním algoritmem vyhledávání je nastíněna struktura diagramů produkovaných nástrojem zvoleným v kapitole 5. Avšak jen v míře nezbytně nutné pro dobré pochopení algoritmu a rozhodnutí, které vedly k jeho zvolení.

Jsou zde také uvedeny postupy použité pro řešení problému se zanořovanáním a překryvem vzorů.

7.1 Popis struktury diagramu

Diagram, který je analyzován ve formě XML souboru, je vytvářen nástrojem *Business Process Visual ARCHITECT 2.4*. Struktura souborů je popsána na následujících řádcích, obrázcích a ukázkách.

Jak je patrné z obrázku 7.1, struktura se drží klasické XML stromové struktury. Kořenovým prvkem je element `Project`, který v sobě obsahuje ty nejpodstatnější informace o verzi programu, exportu, kompatibilitě s verzí *UML*, atd.



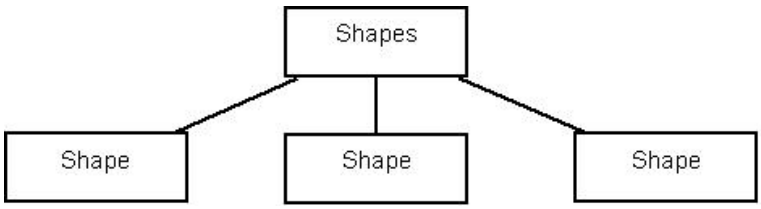
Obrázek 7.1: Stromová struktura uloženého diagramu

Tento element v sobě dále ukrývá tři podstromy: `ProjectInfo`, `Models` a `Diagrams`. Z nich jsou pro mou práci podstatné dva. Těmi jsou `Models` a `Diagrams`.

`Models` samotný je důležitý pro analýzu diagramů, která je prováděna na základě zisku všech jeho elementů a následnou prací s nimi. Struktura `Models` je blíže popsána v části 7.1.1.

`Diagrams` je důležitý pro následné grafické vyznačení jednotlivých nalezených vzorů v rámci diagramu. Podstatný je podstrom `Shapes` a jeho podelementy `Shape`, kdy každý element `Shape` většinou koresponduje s jedním ze zobrazovaných prvků v podstromu `Models`. Jelikož je tato část poměrně implementačně specifická a pro samotnou analýzu nemá příliš velký význam, tak se strukturou, která slouží v programu pro grafické vykreslování, nebudu

dále detailně zabývat. Pouze zmíním, že struktura elementu `Shape` má obdobnou složitost jako struktura `Model` (viz Ukázka 7.1), aby bylo zřejmé, že jeho struktura je poměrně robustní a jeho tvorba s úplnou parametrizací není snadná. Dále ještě naznačím, jak jsou tyto elementy stromově uspořádány (viz Obrázek 7.2).



Obrázek 7.2: Struktura podstromu Shapes

7.1.1 Popis podstromu Models

Popis podstromu `Models` je důležitý pro pochopení následujícího algoritmu pro vyhledávání. V elementu `Models` se nacházejí jednotlivé elementy `Model`. Některé z nich jsou vyobrazovány v diagramu pomocí adekvátních elementů `Shape`, jak bylo uvedeno výše.

Hned na úvod je uvedena ukázka jednoho takového elementu `Model` (viz Ukázka XML 7.1), který je ovšem z důvodů rozsahu zkrácen a jsou zde uvedeny pouze podstatné podelementy, které jsou používány při analýze.

```
<Model composite="false" considerDefaultProperties="false"
   displayModelType="Task" id="hMoXQISGAqACMAx0"
   modelType="BPTask" name="Task17">
  <ModelProperties>
    <StringProperty displayName="Name" name="name"
       value="Task17"/>
    <StringProperty displayName="Model␣Type" name="modelType"
       value="BPTask"/>
    .
    .
    .
    <StringProperty displayName="Id" name="bpmnId"
       value="75"/>
    .
    .
    .
    <DiagramElementRefProperty displayName="Master␣View"
       name="masterView">
      <DiagramElementRef displayShapeType="BPTask"
         id="hMoXQISGAqACMAxz" model="hMoXQISGAqACMAx0"
         name="Task17" shapeType="BPTask"/>
    </DiagramElementRefProperty>
  </ModelProperties>
  <FromSimpleRelationships>
```

```
    <RelationshipRef from="hMoXQISGAqACMAx0"
       id="x5cXQISGAqACMA7q" to="A3kXQISGAqACMA30"/>
  </FromSimpleRelationships>
  <ToSimpleRelationships>
    <RelationshipRef from="vr4XQISGAqACMA0X"
       id="fMsXQISGAqACMA7g" to="hMoXQISGAqACMAx0"/>
  </ToSimpleRelationships>
</Model>
```

Ukázka XML 7.1: Ukázka struktury elementu Model

V ukázce 7.1 je uveden příklad jedné z aktivit (task). Nejdůležitějšími atributy elementu `Model` jsou `id` – obsahující id modelu, `modelType` – typ modelu a `name` – jméno modelu.

Z vnořených elementů jsou nejdůležitější elementy, které obsahují opět jméno a typ prvku. Dále element `DiagramElementRefProperty`, který v sobě ukrývá odkaz na konkrétní element `Shape`, který zajišťuje zobrazení daného modelu.

A poté dva elementy `FromSimpleRelationships` a `ToSimpleRelationships`, které zajišťují odkazy na ostatní prvky, s kterými je daný `Model` provázán.

Důležitou poznámkou je také to, že u bran (Gateway) dochází k zanoření ještě jednoho elementu `Model`. V tomto vnořeném elementu poté bývá specifikováno, o jaký typ brány jde. Mohou se zde nacházet všechny typy uvedené v části 2.2.1. Typ brány je poté specifikován ve vnořeném elementu `Model` v atributu `name`. V tabulce na obrázku 7.3 je uvedeny dvojice *Hodnota atributu name* a *Druh brány*.

Hodnota atributu `name`	Druh brány
XOR	Exclusive Gateway
OR	Inclusive Gateway
AND	Parallel Gateway
Complex	Complex Gateway

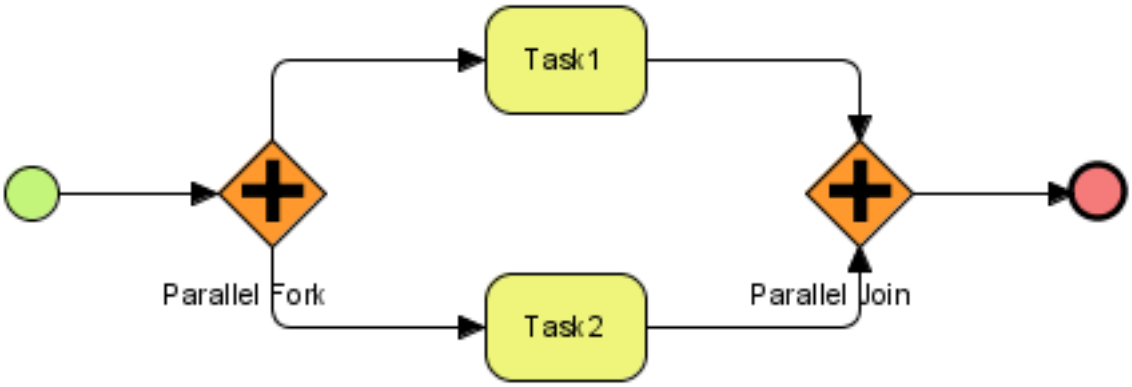
Obrázek 7.3: Kombinace atributu name a typu brány

7.2 Postup při vyhledávání vzorů

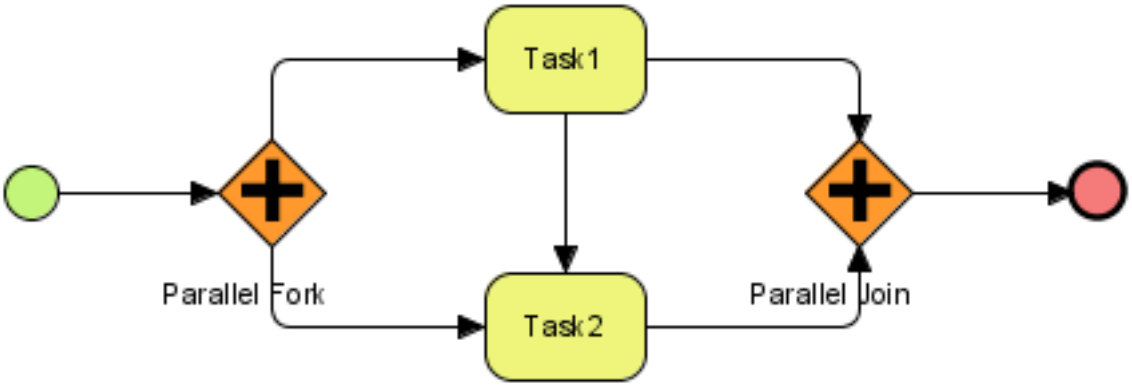
Pro úspěšné vyhledávání vzorů je podstatné zpracování XML souborů s diagramy a také se vzory. Proto je v předchozí části tak důsledně popsána jejich struktura. Je velmi výhodné převést XML stromy do jiných datových struktur, které usnadní zpracování. Není poté třeba pokaždé znova procházet stromovou strukturu, což může být poměrně komplikované a také výpočetně náročnější.

Proto jsou vzory převedeny do datových struktur, které jsou uloženy v polích, s nimiž se pracuje o poznání lépe. Jednotlivé datové struktury odpovídají jednotlivým uzlům v diagramu nebo vzoru, a navíc obsahují odkazy na ostatní připojené prvky.

Vzor se považuje za obsažený v diagramu tehdy, pokud jsou v diagramu všechny patřičné prvky uvedené ve vzoru ve správném pořadí propojení a bez jakýchkoliv propojení navíc. Například tedy správný diagram obsahující vzor *Kombinace Parallel Split a Synchronization* (popsaný v části 3.7.2) je vidět na obrázku 7.4. Naopak diagram neobsahující tento vzor, ačkoliv by to tak na první pohled nemuselo vypadat, je na obrázku 7.5.



Obrázek 7.4: Diagram obsahující vzor Kombinace Parallel Split a Synchronization



Obrázek 7.5: Diagram s porušeným vzorem Kombinace Parallel Split a Synchronization

7.2.1 Nástin algoritmu vyhledávání vzorů

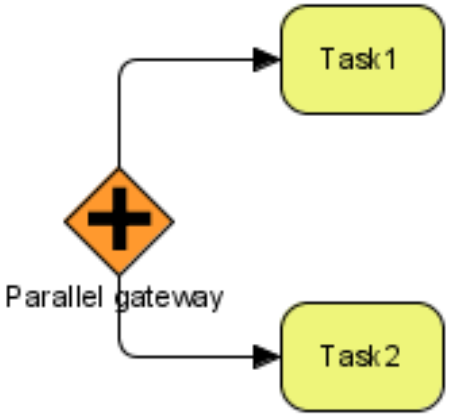
Na počátku vyhledávání je třeba nalézt všechny možné *výchozí body*. Z analýzy totiž vyplynulo poměrně zřejmé pravidlo, že na počátku vzoru se musí nacházet některý z *Flow Objects* (viz část 2.2.1). Na prvním místě tedy musí být událost, aktivita nebo brána. Teprve na tyto atomické uzly se totiž mohou vázat některé z propojovacích prvků.

Za *výchozí body* se tedy dají považovat takové prvky diagramu, které jsou také jedním z množiny – událost, brána, aktivita. A zároveň odpovídá jejich typ prvnímu prvku ve vzoru, jelikož ten obsahuje také jeden z *Flow Objects* jako vstupní bod.

Po určení *výchozích bodů* je možné rekurzivně, nebo nějakým jiným způsobem, dohledávat další cesty v diagramu s pomocí odkazů na další uzly a současným porovnáváním s prvky ve vzoru. Způsob uložení odkazů v rámci diagramů a vzorů je nastíněn v částech 7.1.1 a 6.3.

V případě, že je vyhledáván vzor, ve kterém nejsou využity *zástupné entity*, tak je vyhledávání snadnější a stačí použít rekurzivní algoritmus. Ten vyhledává uzly následující za aktuálním uzlem v modelu a porovnává je s uzly, které se nacházejí ve vzoru. Pokud

uzly ve vzoru a v diagramu odpovídají pro všechny uzly ve vzoru, tak je vzor nalezen. Takovýmto jednoduchým vzorem je například vzor Parallel split (viz část 3.1), který je na obrázku 7.6. Nebezpečím při tomto prohledávání je možnost výskytu cyklu v diagramu. Tuto situaci je z důvodu použití rekurze třeba pečlivě kontrolovat.



Obrázek 7.6: Vzor Parallel Split bez použití zástupných entit

V případě, že je vyhledáván vzor, ve kterém je použita *zástupná entita*, tak je výhodné použít jiné algoritmy pro procházení grafy. Konkrétní použitý algoritmus je popsán v kapitole zabývající se implementací (viz Kapitola 8).

7.2.2 Řešení problémů při vyhledávání

Při vyhledávání vzorů může dojít k několika zásadním problémům:

- Překryv vzorů
- Zanoření vzorů
- Nekorektnost diagramu

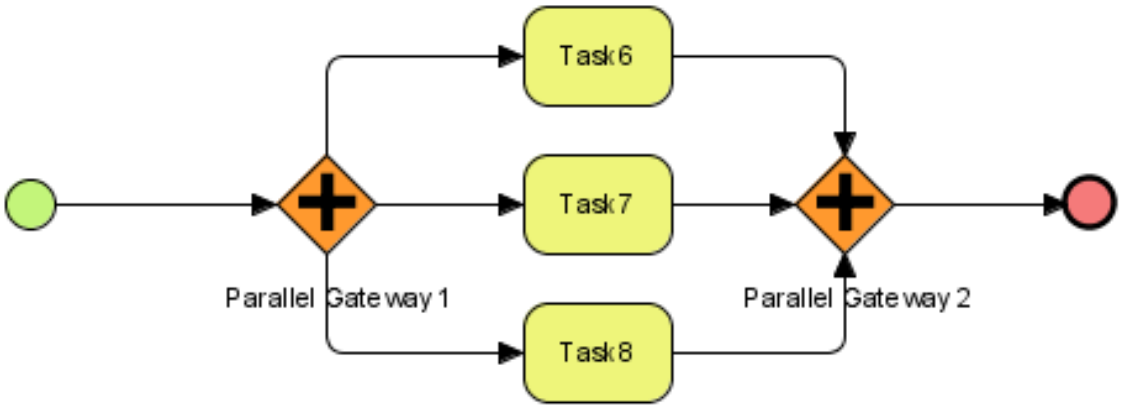
Překryv vzorů – Je situace, kdy více vzorů pro svou existenci v diagramu potřebuje konkrétní uzel nebo množinu uzlů.

K překryvu dochází nejčastěji tak, že je počáteční (výchozí) bod vzoru společný pro více vzorů. Takový případ je možné vidět například na obrázku 7.7. Pokud bychom se v takovémto diagramu snažili nalézt vzor z obrázku 7.6, tak by došlo ke sdílení počátečního uzlu a také části některé z cest vycházejících z brány `Parallel Gateway 1`.

Situace uvedená výše ovšem není jedinou, která by zapříčinila překryv vzorů. Může se stát, že sdílení uzlů vznikne například díky propojení některých cest v diagramu. Této situaci lze částečně předejít tím, že se pro každé větvení a následné spojení toků řízení v diagramu použijí výhradně brány.

Tyto situace lze při vyhledávání detekovat. Například pomocí udržování počtu použití daného uzlu z diagramu ve vzorech.

Jejich řešení ovšem není již natolik jednoduché. Jde především o to, že musí být rozhodnuto, který ze vzorů bude vybrán jako nalezený. Možností je tedy použití jistých preferencí



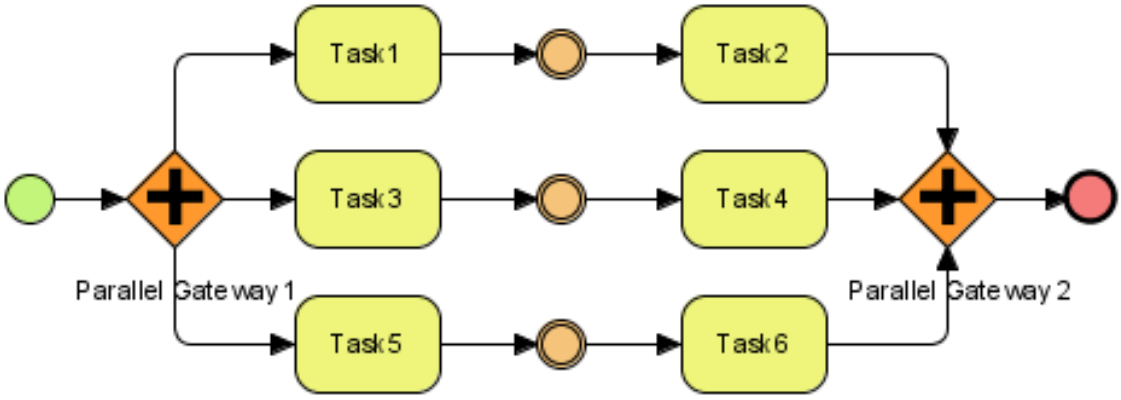
Obrázek 7.7: Ukázka diagramu obsahující překryv vzorů

v rámci tohoto určování. A to například jednoduše pomocí preferencí daných uživatelem, který si zvolí pořadí důležitosti vzorů nebo pomocí porovnání velikosti vzorů na základě počtu obsažených prvků nebo podle prozkoumání, zda vzor není ve více kolizích. Poté by bylo nejlepší odstranit vzor s nejvyšším počtem kolizí.

Zanoření vzorů – Dochází k němu v případě, že vzor v sobě obsahuje jiný vzor. Je to vlastně speciální případ překryvu vzorů, kdy jsou sdíleny všechny uzly ve vzoru jiným vzorem.

Typickým vzorem, který je často zanořen ve vzoru jiném, je *Sequence* (viz část 3.7.1). Tuto situaci je možné vidět na obrázku 7.8. Jde zde vidět, že v *kombinaci Parallel Split a Synchronization* (viz část 3.7.2), která byla výše zvolena za jeden z vyhledávaných vzorů, se nachází jiný vyhledávaný vzor, jímž je *Sequence*.

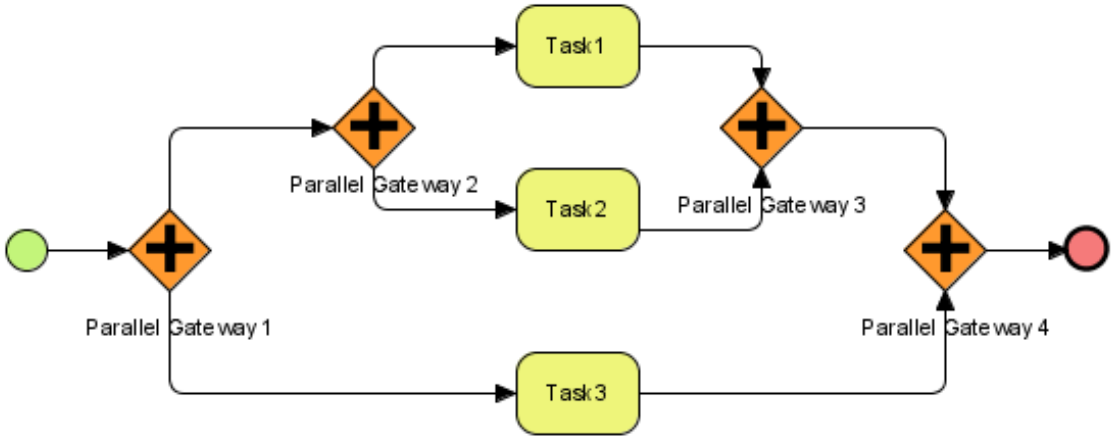
Sequence, která začíná a končí nějakou aktivitou je zde třikrát (každá celá větev mezi bránami). *Kombinace Parallel Split a Synchronization* s použitím zástupné entity *Kombinace větvení a sekvence uzlů* je celý úsek mezi bránami (včetně bran). V takovémto případě je vhodné detekovat oba typy vzorů.



Obrázek 7.8: Ukázka diagramu se zanořením vzorů

Jsou ovšem i případy, kdy detekování zanořených vzorů není zcela vhodné. A v jistých případech lze považovat za poměrně sporné, zda o daný vzor vůbec jde. Takovým případem je například vícenásobné zanoření výše zmiňovaného vzoru *Kombinace Parallel Split a Synchronization* (viz Obrázek 7.9). Problémem je, že se cesta mezi bránami Parallel Gateway 1 a Parallel Gateway 4 opět větví. V literatuře (viz [2], [10]) jsou uvedeny jednoduché vzory bez kombinací nebo kombinace vzorů, které obsahují v průběhu rozvětvených cest pouze aktivity nebo události. V diagramech je poté problém v tom, co je vlastně povoleno mimo přesně vymezené oblasti začátku a konce vzoru.

Bylo zvoleno řešení, kdy v případě přesně definovaných posloupností ve vzorech je akceptován i zanořený vzor. V případě použití *zástupných entit* jsou určovány jen vzory na nižších úrovních. Tedy vzory, které jsou nejvíce zanořené.



Obrázek 7.9: Ukázka diagramu se zanořením větvících se vzorů

Detekce zanoření opět není příliš komplikovaná (vyjma případů zanořování větvení). Řešení tohoto problému je do jisté míry možné pomocí vyhledávání vzoru od rozsahově nejmenších až po největší. Vzory je poté možné jednoduše vypisovat a určovat, zda jsou podstatné nebo ne.

Nekorektnost diagramu – Je situace, kdy nejsou dodržena pravidla pro vytvoření správného diagramu. Takovými záležitostmi jsou například neuvedení počáteční nebo koncové události, jejich použití uprostřed diagramu, atd.

Detekce takovýchto problémů není natolik snadná jako u předchozích dvou problémů. Ale v případě znalosti množiny všech podmínek, které je nutné splnit pro formální správnost diagramu, je možná. Samotná korekce diagramu je ale ve většině případů prakticky neproveditelná. Lze opravit například případy, kdy chybí koncová událost, tím, že ji dodáme. Ale pokud se koncová událost vyskytuje třeba uprostřed diagramu, tak většinou není možné z kontextu určit jaký prvek by zde měl být. Jak jsou tyto situace řešeny v nástroji pro vyhledávání vzorů je uvedeno v kapitole o implementaci (viz Kapitola 8).

7.3 Shrnutí

V této kapitole byla popsána struktura diagramů a v návaznosti na ni byl nastíněn postup vyhledávání vzorů v diagramech. Mimo těchto dvou základních témat, zde byly uvedeny základní problémy, které mohou nastat při vyhledávání vzorů. A byla popsána možná východiska při řešení těchto problémů. Jedním z východisek, jež není uvedeno výše a týká se snížení počtu všech typů kolizí při vyhledávání, je používání jen určité podmnožiny vzorů, které se snažíme detekovat.

Kapitola 8

Popis implementace

V této kapitole je stručně popsána samotná implementace nástroje pro vyhledávání vzorů, který je nazván *BPMN analyzer*. Jsou zde popsány technologie použité pro tvorbu tohoto programu. Jsou tu také uvedeny jeho vlastnosti, přednosti, omezení a řešení jednotlivých situací, u kterých bylo třeba zvolit nějaké konkrétní řešení. Takovým řešením je například vyhledávací algoritmus nebo rozhodnutí o tom jak řešit zanoření nebo překryv vzorů.

8.1 Použité technologie

Program byl především z důvodu přenositelnosti implementován v programovacím jazyce *JAVA*. Konkrétní použitá verze, která byla použita pro testování běhu programu, byla *JRE* 1.6.0_13 [1].

Kromě vlastních zdrojových souborů byla také použita knihovna *JDOM* [2] pro práci s XML soubory, jejich interpretací v paměti počítače a následnou prací s touto strukturou v paměti. Tato knihovna umožňuje jednodušší práci s XML stromem než jakou poskytuje nativní přístup v jazyce *JAVA*.

8.2 Vyhledávací algoritmus

Jak již bylo zmíněno v části 7.2, tak je problém vyhledávání rozdělen na dva případy. A to podle toho, zda obsahuje hledaný vzor nějakou zástupnou entitu, nebo ne.

V obou případech algoritmus začíná tím, že se naleznou *výchozí body*. Z těchto jednotlivých bodů je pak postupně prováděno následné dohledávání vzorů.

Samotné vyhledání výchozích bodů probíhá tak, že je použit první uzel ve vzoru jako referenční a všechny uzly, které mu v diagramu typem odpovídají, jsou následně použity jako výchozí body.

V následných dvou částech je popsán zbytek dohledávání vzorů odděleně pro přesně dané vzory a vzory se zástupnými entitami.

8.2.1 Vyhledávání přesně daných vzorů

Jak již bylo uvedeno v předešlé kapitole, tak je pro vyhledávání výhodné použít rekurzivní algoritmus, který na základě daného vzoru nalezne uzly v diagramu, jež do tohoto vzoru

[1] Aktuální verze je dostupná na adrese http://www.java.com/en/download/

[2] Je dostupná na adrese http://www.jdom.org/

mohou patřit.

Algoritmus postupuje tak, že v prvním kroku vezme jako aktuální prvky výchozí bod nalezený v diagramu a první prvek ve vzoru.

Následně bere jednotlivé větve vycházející z uzlu ve vzoru zakončené dalším uzlem a snaží se tyto uzly přiřazovat jednotlivým prvkům na koncích hran v diagramu.

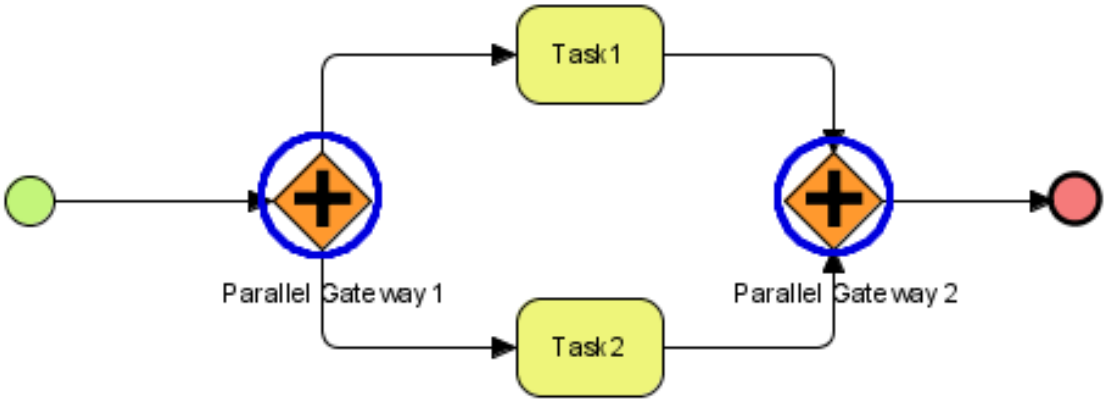
Pokud se podaří přiřadit uzel ve vzoru, který vychází z aktuálního prvku, některému prvku v diagramu, tak se tento stav zaznamená, uzly se označí jako použité a stav je použit jako další kombinace, z které se dá vycházet pro vyhledávání. Stavy jsou průběžně udržovány v poli a v případě nemožnosti rozvíjení aktuálního stavu je snaha rozvinout další kombinaci.

Prohledávání končí pro daný stav úspěšně v případě, že není žádný další uzel ve vzoru a všechny uzly jsou přiřazené. Pokud prohledávání skončí úspěšně pro jakýkoliv stav, tak je úspěšné i celkově.

Prohledávání končí pro daný stav neúspěšně, pokud není žádný další uzel v diagramu, který by šlo použít nebo pokud si uzly ve vzoru a diagramu neodpovídají v žádné povolené kombinaci.

Celé toto prohledávání je jistou variací na metodu procházení grafů, kterou je *Backtracking*.

Pro lepší vysvětlení algoritmu následuje příklad, kde je nastíněno vyhledávání na již v předchozích kapitolách zmiňovaném vzoru *Parallel Split* (viz Obrázek 7.6). Ten se budeme snažit najít v diagramu na obrázku 8.1. Na obrázku jsou modře vyznačeny uzly, které jsou na počátku určeny jako výchozí.



Obrázek 8.1: Diagram pro vyhledání Parallel Split

Vyhledávání začne tak, že se jako aktuální prvek nastaví `Parallel Gateway 1` v diagramu a ve vzoru se jím stává `Parallel Gateway`. Ve vzoru se vezme `Task1` a označí se za aktuální prvek, který má být vyhledáván v diagramu.

Adekvátně k němu je nalezen `Task1` v diagramu. Tato dvojice je uložena jako stav pro rozgenerování. Protože ve vzoru z aktuálního prvku nevede žádná další cesta, tak je za aktuální hledaný prvek označen `Task2`.

Vzhledem k tomu, že je uzel `Task1` v diagramu označen za použitý, tak je nalezen jako první adekvátní prvek v diagramu `Task2`.

Pokud z aktuálního prvku ve vzoru ani z žádného předchozího uzlu nevedou žádné cesty, které by nebyly nalezeny, tak je vzor prohlášen za nalezený v daném stavu s pokrytím uzlů

`Parallel Gateway 1`, `Task1`, `Task2`.

Následně se algoritmus snaží hledat z výchozího bodu `Parallel Gateway 2`, ale zde nedojde k žádné shodě mezi vzorem a diagramem. Pak je toto hledání ukončeno jako neúspěšné.

8.2.2 Vyhledávání vzorů se zástupnými entitami

Zástupné entity byly popsány v části 6.1.1. Jejich využití umožňuje vyhledávání i takových vzorů, které nejsou zcela přesně specifikovány. To znamená, že není zcela jednoznačně popsána posloupnost, v jaké musejí prvky za sebou následovat a jaký musí být jejich typ a počet. Na jedné straně toto přináší jistou volnost v popisu vzorů, ale na druhé straně tento přístup má i určitá úskalí. Ta budou podrobněji rozebrána v části 8.4.

Samotný algoritmus pro vyhledávání takovýchto vzorů má s předchozím způsobem společný postup pro nalezení výchozích bodů.

Po nalezení výchozích bodů dochází k nastavení aktuálního uzlu v diagramu. Poté se prochází diagramem tak dlouho, dokud není nalezen konec diagramu nebo ukončující prvek pro vyhledávání, což bývá pro zástupný element *Sekvence uzlů* takový prvek, který se nachází ve vzoru těsně za tímto zástupným elementem. Obdobně tomu je u zbývajících zástupných elementů. U těchto je předpokládán jako následný prvek nějaký typ brány.

Jakmile je nalezen tento počáteční a koncový uzel, tak se prochází diagramem postupně pomocí upraveného algoritmu *Prohledávání do šířky*. Jednotlivé prvky jsou postupně ukládány do pole výsledných uzlů a přitom je kontrolováno, zda se na cestě neobjeví nějaká neočekávaná brána, díky které by algoritmus byl ukončen.

8.3 Řešení kolizí vzorů

Řešením kolizí je v tomto případě řešení problémů se zanořením a překryvem vzorů.

Jelikož nejsou zaimplementovány žádné preference uživatele a je snaha o zachování maximální obecnosti vyhledávání, jsou detekovány všechny možné vzory.

Jestliže se tedy dva různé vzory překrývají, jsou detekovány oba. Pokud dochází k překryvu jednoho vzoru, například jeden vzor začíná dvakrát v jednom výchozím bodu, tak je zaznamenán pouze první nalezený vzor.

V případě zanoření vzorů do sebe jsou za výsledek také považovány všechny vzory. Problém nastává až v případě výskytu bran uvnitř zanoření a vyhledávání vzorů se zástupnými entitami. V takovém případě nemusí dojít ke zcela korektnímu nalezení vzoru na vyšších úrovních. Tento problém byl diskutován už v části 7.2.2.

8.4 Vlastnosti programu

První důležitou připomínkou je, že nástroj předpokládá pouze vytváření menších vzorů, které neobsahují více než 20 prvků. Ani v literatuře se téměř nevyskytují vzory s vyšším počtem uzlů. A lze tedy předpokládat, že v případě objevení obdobně rozsáhlého vzoru došlo k nedobrému rozkladu problému a zachycení opakující se situace (tedy vzoru). Je také vysoká pravděpodobnost, že velký vzor bude obsahovat velké množství podvzorů, které by výstup programu výrazně znepřehlednily a docházelo by k velkému množství kolizí vzorů při vyhledávání. Program tedy nemá problém se samotným vyhledáváním velkých vzorů, ale dochází ke značně nepřehledným situacím v případě vytváření velkých vzorů a prohlížení

výstupních upravených diagramů, v nichž byly tyto velké vzory vyhledávány. Velikost prohledávaných diagramů však není omezena.

Vzory k vyhledávání lze vytvářet v editoru, který je součásti implementovaného nástroje, a to poměrně jednoduše pomocí výběru jednotlivých komponent, z kterých se má vzor skládat, a jejich následného propojování. Z editoru lze posléze ukládat vzory do XML souborů, které lze naimportovat do *BPMN analyzeru* pro vyhledávání.

Při tvorbě vzorů se zástupnými entitami je třeba mít na paměti, že je nutné stanovit, jaké prvky budou těsně navazovat na zástupnou entitu.

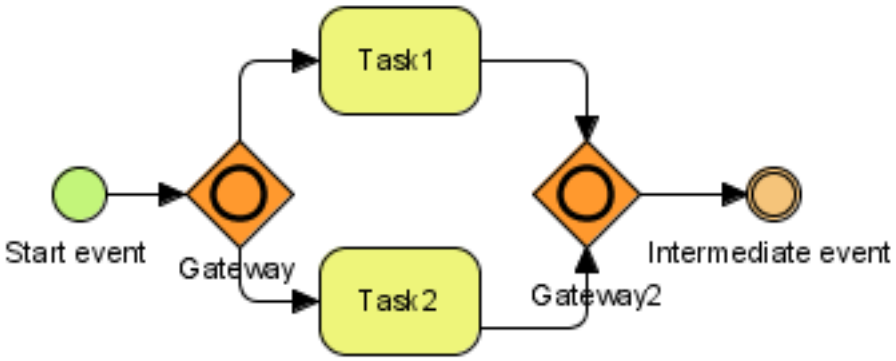
Zástupné entity jsou obecně určitou abstrakcí problému, která je jen jedním z možných přístupů k problému. Namísto nich mohl být použit například postup, který nalezne počátek a konec kombinace vzorů a celý stavový prostor mezi nimi se prohlásí za součást této kombinace vzorů. Problémy by ovšem nastávaly v případě, že by se některá z cest v této části větvila a dosahovala určitou „oklikou“ za koncový bod vzoru.

Po důkladném zvážení byly tedy zvoleny zástupné entity s určitými omezeními v diagramu jako vhodné řešení. Omezením v tomto směru je myšlena nevhodnost obsažení bran v diagramu v části, která by měla být pokryta zástupnou entitou, a větvení diagramu jinde než v místě bran.

Omezením týkajícím se vyhledávání je skutečnost, že program nerozeznává jednotlivé typy událostí. To znamená, že počáteční události, události v průběhu a koncové události pro něj mají stejnou váhu. Pro samotnou detekci tato skutečnost má význam v případě, že diagram je zapsán nekorektním způsobem a na určitém místě v diagramu se vyskytne neočekávaná událost. Poté tato skutečnost není zaznamenána a může dojít k pokračování v hledání vzoru. Příkladem může být umístění události `Intermediate event`, která patří do průběhu vnitřní cesty , na konec diagramu (viz Obrázek 8.2).

Velkou část situací, jež by mohly vést ke vzniku nevalidního diagramu, kontroluje samotný nástroj pro jejich tvorbu (*Business Process Visual ARCHITECT 2.4*), který neumožňuje například vedení toku událostí ve směru do počáteční události nebo ve směru z koncové události, atd.

Situace, kdy se v diagramu vyskytují podprocesy, je dalším omezením pro vyhledávání. Nástroj nepředpokládá, že se podprocesy budou v diagramu vyskytovat a s vysokou pravděpodobností by v takovémto diagramu nenašel hledaný vzor, jehož součástí by podproces byl.



Obrázek 8.2: Příklad nevalidního diagramu

8.5 Shrnutí

V rámci této kapitoly byly popsány zvolené vyhledávací algoritmy, byla uvedena některá omezení programu a objasněna řešení překryvu a zanoření vzorů.

Celkově aplikace *BPMN analyzer* splňuje požadavky, které jsou na ni kladeny. Což znamená, že dokáže zpracovat diagramy vytvořené v nástroji *Business Process Visual ARCHITECT 2.4*, jež jsou do vytvořené aplikace nahrány ve formě XML souborů. Diagramy jsou posléze prohledány na výskyt vzorů. Tyto vzory je možné v rámci aplikace také vytvářet. *BPMN analyzer* je tedy schopen vyhledávat prakticky obecné vzory. Obecnost vzorů je podpořena využitím *zástupných entit*, které byly popsány výše.

Kapitola 9

Závěr

Během stále se rozvíjejícího využívání modelování byznys procesů se standardem pro jejich prezentaci stala notace *BPMN*.

A vzhledem k tomu, jak roste počet existujích diagramů, také stoupá zájem o možnosti jejich zpracování, analyzování, kontroly, atd. K tomu také přispívá zvyšující se složitost zaznamenávaných procesů, jelikož notace umožňuje sestavovat hierarchickou strukturu procesů tak, že je nimi možné zachytit fungování celého systému.

Tento trend je reflektován v této práci vytvořením nástroje, jenž dokáže diagramy analyzovat z pohledu výskytu vzorů. Tyto vzory vypovídají o určité situaci, k níž v procesu dochází. Pro tyto situace jsou většinou již předpřipravená řešení, která dokáží tuto situaci zefektivnit nebo jiným způsobem ošetřit.

Nástroj je schopen vyhledávat přesně specifikované vzory. Ale je do něj také zanesena možnost vytvoření vzorů poskytujících určitou volnost v jejich tvorbě. Tuto svobodu přinášejí *zástupné entity*, prvky diskutované v kapitole 6. Jejich přínos je především ve schopnosti zachování ne zcela přesné definice vzoru. Není tedy nutné popsat každý prvek, ale je možné určitou posloupnost uzlů ve vzoru nahradit touto zástupnou entitou.

Tato výhoda je ovšem zmenšena jistým omezením. Tím je nutnost dodržovat při tvorbě diagramu určitá pravidla. Nejvýznamnějším z nich je podmínka, že k větvení a spojování cest v diagramu dochází pouze v branách. Druhé omezení má také souvislost se zástupnými entitami, ale také s problémem kolizí vzorů. Je jím problém s výskytem bran na cestě, která má být pokryta zástupnou entitou.

Odstranění těchto dvou limitujících faktorů je jednou z dobrých možností rozšíření systému.

Další možností úpravy systému je změna přístupu k řešení kolizí vzorů. V aktuální implementaci jsou řešeny kolize a zanoření tak, aby byl nalezen a vypsán maximální počet detekovaných vzorů. A to i takových, které jsou v kolizi nebo jsou zanořeny. Možná východiska řešení jako jsou *preferenční vyhledávání vzorů*, *výpis vzorů s minimálním počtem kolizí*, atd. jsou nastíněna v kapitole 7.

Z pohledu zadání diplomové práce byly splněny všechny požadavky, které byly na tuto práci kladené. Počínaje prostudováním notace BPMN a vzorů, jež se v diagramech byznys procesů mohou vyskytovat.

Následně byla provedena volba nástroje. Tímto nástrojem se stal *Business Process Visual ARCHITECT* v dané chvíli konkrétně ve verzi 2.4. Volba nástroje musela být provedena, protože i přes existující standardní formát se stále některé aplikace pro tvorbu diagramů v některých aspektech z těchto pravidel vymykají. Možným rozšířením práce v tomto směru by bylo použití obecného formátu ve chvíli, kdy všechny modelovací nástroje na tento

formát přejdou a budou jej beze zbytku dodržovat.

Také byl vyřešen problém s tvorbou obecně zapsaných a vyhledávaných vzorů, s nimiž aplikace dokáže pracovat. Na jejich základě byl navržen postup pro vyhledávání vzorů v diagramech, který je konkrétně popsán v části 8.2. Na tomto místě byla také popsána omezení při vyhledávání, jež by mohla být v rámci další práce odstraněna.

Celkový vývoj a rozšiřování použití modelování byznys procesů může společně s jejich analýzou vést ke značnému zvýšení efektivnosti samotných procesů v organizacích. Tím snížit náklady a zvýšit konkurenceschopnost. Zlepší se tím také šance organizace na udržení dobře optimalizovaných procesů. A to i v případě vyšší fluktuace zaměstnanců a současné složité situaci na trhu.

Literatura

[1] van der Aalst, W. M. P.: Patterns and XPDL: A Critical Evaluation of the XML Process Definition Language. [online] `http://is.tm.tue.nl/research/patterns/download/ce-xpdl.pdf`.

[2] van der Aalst, W. M. P.; Barros, A. P.; ter Hofstede, A. H. M.; aj.: Advanced Workflow Patterns. [online] `http://www.fit.vutbr.cz/~weiss/dokuwiki/lib/exe/fetch.php?id=publikace:publikace&cache=cache&media=publikace:seminar_uifs_20071119.pdf`, 2000.

[3] van der Aalst, W. M. P.; ter Hofstede, A. H. M.; Kiepuszewski, B.; aj.: Workflow Patterns. [online] `http://www.workflowpatterns.com/documentation/documents/wfs-pat-2002.pdf`.

[4] van der Aalst, W. M. P.; Russell, N.; ter Hofstede, A. H. M.; aj.: WORKFLOW CONTROL-FLOW PATTERNS: A Revised View. [online] `http://www.workflowpatterns.com/documentation/documents/BPM-06-22.pdf`.

[5] Demel, J.: *Grafy a jejich aplikace*. Praha: Academia, 2002, iSBN 80-200-0990-6.

[6] Harold, E. R.; Means, W. S.: *XML v kostce – Pohotová referenční příručka*. Praha: Computer Press, 2002, iSBN 80-7226-712-4.

[7] Object Management Group: Business Process Modeling Notation Specification, V1.1. [online] `http://www.omg.org/spec/BPMN/1.1/PDF`, 2008.

[8] W3C: Extensible Markup Language (XML) 1.1 (Second Edition). [online] `http://www.w3.org/TR/2006/REC-xml11-20060816`, 2006.

[9] White, S. A.: Introduction to BPMN. [online] `http://www.bpmn.org/Documents/Introduction to BPMN.pdf`.

[10] White, S. A.: Process Modeling Notations and Workflow Patterns. [online] `http://www.bptrends.com/deliver_file.cfm?fileType=publication&fileName=03-04 WP Notations and Workflow Patterns - White.pdf`, únor 2008.

[11] Wohlgemuth, J.: *Bakalářská práce - Transformace byznys procesů*. Brno: FIT VUT v Brně, 2008.

Dodatek A

Obsah CD

Obsah přiloženého CD:

- text této diplomové práce ve formátu pdf a ve formě zdrojových souborů pro LaTeX
- zdrojové soubory programu
- uživatelský manuál
- soubor readme.txt s popisným seznamem obsahu CD